UNIVERSIDAD CARLOS III DE MADRID

Escuela Politécnica Superior - Leganés

INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

APLICACIÓN DE TÉCNICAS DE CLUSTERING PARA LA MEJORA DEL APRENDIZAJE

AUTOR: MARGARITA GALLARDO CAMPOS
TUTOR: ANTONIO JESÚS DE CASTRO GONZÁLEZ
DIRECTOR: ESTEBAN GARCÍA CUESTA

Leganés, 2009

TÍTULO: *Aplicación de técnicas de clustering para la mejora del aprendizaje*.

AUTOR: Margarita Gallardo Campos

TUTOR: Antonio Jesús de Castro González

DIRECTOR: Esteban García Cuesta

La defensa del presente Proyecto Fin de Carrera se realizó el día 29 de Mayo de 2009; siendo calificada por el siguiente tribunal:

PRESIDENTE: Fernando López Martínez

SECRETARIO: Jose Ramón Martín Solís

VOCAL: Jose Antonio Iglesias Martínez

Habiendo obtenido la siguiente calificación:

CALIFICACIÓN:

Presidente **Secretario** **Vocal**

A mi madre,
a mi abuelo,
a Brian,
y a todos los que creyeron en mí.

Agradecimientos.

Llegado este momento tan ansiado, con el que tantas veces he soñado y que parecía imposible que llegara, sólo me queda agradecer:

A Esteban García Cuesta, por introducirme en el campo del aprendizaje automático, por lo mucho que he aprendido a su lado, por todo lo que me ha ayudado, por su paciencia y por prestarme tanta atención como yo necesitaba.

A Antonio Jesús de Castro González, por brindarme la oportunidad de trabajar con él y por presentarme a Esteban. A la gente del LIR, por abrirme un hueco en su laboratorio y por haber sido tan agradables conmigo, haciéndome sentir como si los conociera de toda la vida.

A los amigos que me llevo de esta universidad. Por todas las cosas que hemos vivido juntos, los momentos de tensión por esas prácticas que nunca terminaban, por los nervios de los exámenes, por las risas en los descansos, por los momentos en la cafetería, por aguantar mi mal humor y mis momentos de estrés y, en definitiva, por haber conseguido que disfrutara tanto de esta carrera. Gracias por vuestra paciencia, sin vosotros no hubiese conseguido estar aquí.

A mis amigos del parque, quienes, después de tanto tiempo, siguen queriéndome tal como soy y en todo momento. Por sus ánimos, por escucharme, por estar siempre ahí cuando los necesitaba. Por enseñarme a que hay que disfrutar de las pequeñas cosas que te da la vida. Gracias chicos.

A Sofía y a Esther, por su alegría y su gran amistad.

A Brian, por darme todo su amor en estos últimos años y por quererme tanto, por su paciencia, su comprensión, su entereza a veces envidiable y por enseñarme a ser fuerte y confiar en mí misma.

A mi familia en general, porque ellos nunca han dudado de mí y siempre han sabido que yo podía con todo aquello que me propusiera. Y a mi periodista favorita, María, por ayudarme a repasar la estructura de la memoria en los últimos días.

A mi abuelo, por ser el mejor padre.

A mi madre, para quien no existen suficientes palabras para agradecer todo lo que ha hecho por mi. Gracias por tanto amor, apoyo, abnegación, por la educación que me has dado, por

los valores que me has transmitido y, sobretodo, por todos los esfuerzos y sacrificios que has realizado para que yo llegue hasta aquí, sin esperar nunca nada a cambio. Lo conseguí. Espero que te sientas orgullosa, y que a partir de ahora respires un poco más tranquila.

Muchas gracias a todos.

Marga

El atractivo del conocimiento sería muy pequeño si en el camino que conduce a él no hubiera que superar tanto pudor.
Nietzsche

Lo poco que sé, se lo debo a mi ignorancia.
Platón

Resumen.

El aprendizaje automático es una rama de la inteligencia artificial, y se puede definir como el conjunto de técnicas, métodos y sus implementaciones algorítmicas capaces de aprender y mejorar su eficacia a través de la experiencia. En la mayoría de los casos se busca realizar ese aprendizaje a partir de información no estructurada y sin supervisión humana. En las últimas décadas, el uso de técnicas de aprendizaje automático en aplicaciones en campos tan diversos como la informática, la estadística, la robótica, la medicina, etc. se ha visto incrementado de manera extraordinaria. Dada esta gran demanda de aplicaciones, el desarrollo de nuevas técnicas de aprendizaje automático también se ha visto incrementado de manera notable.

Uno de los objetivos principales de este Proyecto de Fin de Carrera es el de presentar el estado del arte del área de aprendizaje automático. En esta memoria se repasarán brevemente las aplicaciones y la taxonomía clásica de las distintas técnicas de aprendizaje automático. De entre éstas, se hará especial hincapié en las técnicas de aprendizaje automático no supervisadas, concretamente, en las técnicas de agrupamiento o *clustering*.

El agrupamiento o *clustering* consiste en la clasificación de datos, observaciones o vectores de características en grupos (*clusters*), sin tener ningún tipo de información sobre la salida. El resultado de un agrupamiento es un conjunto de grupos en los cuales los objetos de un mismo grupo son más similares entre sí que con objetos de otros grupos. Con este trabajo se pretende realizar un repaso a los distintos tipos de algoritmos de agrupamiento y algunas de sus aplicaciones.

Uno de los tipos de algoritmos de agrupamiento más recientes son los denominados algoritmos de agrupamiento basados en densidades. Estas técnicas intentan resolver algunos de los problemas que presentan los algoritmos de agrupamiento tradicionales. El presente proyecto, realiza una breve introducción a este tipo de técnicas de agrupamiento basadas en densidades. Posteriormente, se explican detalladamente tres algoritmos dentro de este tipo. Estos son: DBSCAN, *Mean Shift Clustering* y un nuevo algoritmo de agrupamiento denominado LPC.

El último objetivo de este Proyecto de Fin de Carrera es realizar un estudio comparativo de los tres algoritmos de agrupamiento basados en densidades, DBSCAN, Mean Shift y LPC, para así poder comprender mejor las características cualitativas de las agrupaciones basadas en densidades y medir cuantitativamente la relación existente entre estos tres algoritmos.

Abstract.

Machine learning is a subfield of artificial intelligence and it can be defined as the set of techniques, methods and algorithms which are capable of learn and improve its efficiency throughout experience. The aim is to learn from data without structure and, in most cases, without human supervision. During the last decades, the use of machine learning techniques in applications in fields as wide as informatics, statistics, robotics, medicine, etc. have been increasing incredibly. Due to the huge demand in these applications, there has been an increase in the development of new machine learning techniques.

One of the main goals of this Final Project is to present the state of art of machine learning. Throughout this work a brief survey of the applications and a classic taxonomy of the different machine learning techniques will be carried out. Among these, special interest will be paid to non-supervised machine learning techniques, specifically, clustering techniques.

Clustering is the way to classify data, observations or feature vectors in clusters without any information about the outputs. The results are clusters or groups on which objects belonging to the same group are more similar among them than among objects from other groups. Throughout this work a review of different clustering algorithms and some of its applications is done.

Nowadays, one of the most recent clustering techniques are the density based clustering techniques. These techniques try to solve some of the problems that traditional clustering algorithms have. This Final Project makes a brief introduction to these types of density based clustering techniques. Subsequently, a detailed explanation of three algorithms, based on the idea of density based clustering, will be given. These algorithms are: DBSCAN, Mean Shift Clustering and a new clustering algorithm denominated LPC.

The last goal of this Final Project is to perform a comparison of these three density based algorithms: DBSCAN, Mean Shift Clustering and LPC, so as to comprehend easily the qualitative features of the density based groups and qualitatively measure the relations existing among these three algorithms.

Índice general.

Índice de figuras.

Índice de tablas.

Índice de algoritmos.

Capítulo 1

Introducción y objetivos.

Este primer capítulo está dedicado a proporcionar una visión global del contenido de este Proyecto de Fin de Carrera. En primer lugar, se analizarán las motivaciones que han llevado a emprender este trabajo. A continuación, se presentan los principales objetivos que se pretenden alcanzar con realización de este proyecto. Para terminar el capítulo, se describirá brevemente la organización y el contenido de esta memoria.

1.1. Motivación.

La utilización de técnicas de agrupamiento (*clustering*) ha sido ampliamente tratada en diversos campos. El agrupamiento es una técnica común en el área de análisis estadístico y se ha ido extendiendo, por su intrínseca aplicabilidad, a otros campos como el reconocimiento de patrones, minería de datos, análisis de imágenes o aprendizaje automático.

Tradicionalmente, las técnicas de agrupamiento se han clasificado en dos tipos fundamentales: las técnicas jerárquicas y las particionales. Las primeras se corresponden con análisis recursivos de agrupamiento en los datos para ir obteniendo de una manera paulatina una jerarquía de pertenencia de grupos. En la segunda clasificación, se pretende obtener un único nivel de subconjuntos, donde cada uno de ellos recoge un comportamiento homogéneo con respecto al conjunto total.

Recientemente, han surgido nuevos métodos que, por su relevancia, no son incluidos dentro de la clasificación previa, sino que han dado lugar a una nueva clasificación con nuevos tipos. Un ejemplo de estos nuevos tipos son los denominados métodos probabilísticos. Estos métodos asumen que los grupos están formados por objetos generados por distintas distribuciones de probabilidad, de modo que el agrupamiento se realiza en base a la suposición de una serie de parámetros probabilísticos.

Otro de los tipos de algoritmos de agrupamiento que han obtenido gran relevancia en los últimos años son los algoritmos de agrupamiento basados en densidades. En estos algoritmos, los grupos son considerados como regiones densas en el espacio de datos y se encuentran separados entre sí por regiones de baja densidad de objetos (ruido). Una característica de este tipo de algoritmos es que son capaces de encontrar grupos de formas arbitrarias y pueden estar distribuidos de cualquier manera.

Según los datos de los que disponemos, no existen actualmente muchas comparativas de estos métodos de agrupamiento, por lo que debido a su novedad y su carácter aplicado (por ejemplo, selección estable de características en espacios de alta dimensionalidad [1]), la realización de un estudio más profundo de las características de estos métodos puede considerarse como un trabajo de generación de conocimiento necesario para la búsqueda de nuevas aplicaciones y algoritmos.

Este trabajo está orientado desde el punto de vista del aprendizaje automático y, por tanto, utilizaremos terminología e historia referente a esta área y no a ninguna de las otras áreas mencionadas anteriormente. Aunque no se tratará la terminología de las otras áreas, tampoco trataremos de obviar ninguno de los puntos que puedan tener en común.

1.2. Objetivos.

El objetivo principal de este Proyecto de Fin de Carrera es el de establecer el estado del arte y la realización de un estudio comparativo de las técnicas de agrupamiento basadas en densidades dentro del campo de aprendizaje automático.

Para ello, será necesario realizar una introducción al campo de aprendizaje automático y a las técnicas de agrupamiento para, posteriormente, profundizar en las técnicas de agrupamiento basadas en densidades. Se explicarán en detalle tres algoritmos dentro de esta tipología. Estos son: DBSCAN (*Density-Based Spatial Clustering of Applications with Noise* [2]), el agrupamiento Mean Shift [3] y un nueva técnica de agrupamiento denominada LPC basada en un algoritmo de búsqueda de curvas locales principales definido por Einbeck et al. en [4].

Los requisitos deseados por cualquier algoritmo de agrupamiento se pueden resumir entre otros en: la escalabilidad, la posibilidad de tratar con distintos tipos de atributos, la posibilidad de descubrir grupos con distinta forma, la capacidad de detectar ruido o *outliers*, la capacidad de manejar datos de gran dimensionalidad, la facilidad de interpretación o la posibilidad de tener un conocimiento mínimo del entorno para determinar los parámetros de entrada. En este proyecto de fin de carrera se desarrollará una comparativa de estas tres técnicas de agrupamiento para poder obtener conclusiones sobre su comportamiento frente a estos requisitos deseables.

Las tres técnicas elegidas están fundamentadas en diferentes bases argumentales y matemáticas, por lo que la comparativa recogerá de manera extensa el estado actual de las técnicas de agrupación basadas en densidades facilitando de este modo su comprensión y entendimiento.

1.3. Estructura de la memoria.

La organización del contenido de la memoria del presente Proyecto Fin de Carrera se ha llevado a cabo de la siguiente manera:

- **Capítulo 1: Introducción.**

  A lo largo de este capítulo se ofrece una visión global del proyecto, se explica la motivación de su realización, se presentan los objetivos del mismo y se describe la organización de la memoria.

- **Capítulo 2: Estado del Arte.**

  El objetivo principal de este capítulo es el de presentar el estado del arte del área de aprendizaje automático como marco general de las técnicas de agrupamiento que serán motivo de comparación en el presente proyecto.

  El capítulo esta estructurado en dos partes fundamentales. La primera parte ofrece una visión global del aprendizaje automático. En ella se realiza una breve taxonomía, se explican algunas de las técnicas comúnmente utilizadas y se introducen algunas de las aplicaciones del aprendizaje automático.

  En la segunda parte de este capítulo, se profundiza en el aprendizaje automático no supervisado y más concretamente en las técnicas de agrupamiento (o, en inglés, *clustering*). En ella se lleva a cabo una breve introducción explicando la motivación de este tipo de técnicas de aprendizaje no supervisado. Después, se explican las distintas medidas de similitud o de distancia necesarias para su implementación y se realiza una visión general sobre los distintos tipos de algoritmos de agrupamiento existentes. Finalmente, se presentan las distintas aplicaciones de los algoritmos de agrupamiento.

- **Capítulo 3: Algoritmos de agrupamiento basados en densidades.**

  En este capítulo, tras introducir brevemente algunos de los algoritmos de agrupamiento más comunes, se presentan un nuevo tipo de algoritmos de agrupamiento basados en densidades que dan solución a algunos de los problemas que poseen los algoritmos de agrupamiento clásicos.

  En este capítulo se profundiza en las técnicas de agrupamiento basadas en densidades, analizando en que consisten y los principales algoritmos implementados hasta hoy en día. Posteriormente, se estudian en profundidad el algoritmo DBSCAN y una versión de este algoritmo denominado DBSCAN-4C. A continuación, se estudia el algoritmo de agrupamiento *mean shift clustering* basado en el desplazamiento de medias y más reciente que el anterior. Por último, se define la implementación de un nuevo algoritmo de agrupamiento basado en la búsqueda de curvas locales principales denominado agrupamiento LPC.

- **Capítulo 4: Comparación de los algoritmos DBSCAN, Mean Shift Clustering y LPC.**

En el capítulo anterior se estudian las características principales de los tres algoritmos de agrupamiento basados en densidades, DBSCAN, Mean Shift y LPC, con la intención de poder comprender su implementación. Sin embargo, en ese capítulo no se aprecia la relación que existe entre estos tres algoritmos de agrupamiento basados en densidades. Con el fin de comprender mejor sus características generales y sus ventajas e inconvenientes, en el presente capítulo se lleva a cabo una comparación de los tres algoritmos nombrados anteriormente, de manera que se facilite su interpretación, su comprensión y su aplicabilidad en futuros nuevos problemas.

- **Capítulo 5: Conclusiones.**

  Este último capítulo se presentan las conclusiones finales consecuencia del trabajo realizado y se discuten los resultados obtenidos.

- **Apéndice A: Algoritmos implementados en Matlab.**

  Este apéndice incluye los códigos implementados en Matlab de los tres algoritmos explicados en este proyecto. El código se presenta documentado (en inglés) y se presenta un ejemplo de traza de ejecución.

Capítulo 2

Estado del arte.

La posesión de información y la posterior adquisición del conocimiento adecuado a cada situación han jugado un papel trascendental en la historia del hombre.

Desde el comienzo de la vida, la sabiduría y la posesión de información han sido sinónimos de poder y de supervivencia. En nuestros días, la cantidad de información de la que disponemos es enorme haciendo complicado su manejo y su tratamiento. La introducción de la informática y las nuevas tecnologías han conseguido no sólo que se puedan obtener y almacenar grandes cantidades de datos, sino que además nos preguntemos si es posible un uso inteligente de estos. De este modo, se intentan desarrollar métodos de análisis de manera que los ordenadores sean capaces de aprender sobre estos datos, mejorar la eficiencia de sus aplicaciones y obtener algún conocimiento sin apenas intervención humana.

Como consecuencia de estas necesidades surge una nueva área de investigación y trabajo denominado aprendizaje automático (*machine learning*), cuyo crecimiento en las últimas décadas ha sido extraordinario.

Uno de los objetivos principales de este capítulo es el de presentar el estado del arte del área de aprendizaje automático como marco general de las técnicas de agrupamiento o *clustering* que serán motivo de comparación en el presente proyecto. La estructura de este capítulo se puede dividir en dos partes fundamentales. En la primera parte de este capítulo, se ofrece una visión general del significado del aprendizaje automático. Además, se detallan algunas de las técnicas más comunes en el aprendizaje automático, haciendo una breve taxonomía e introduciendo algunas de las aplicaciones para las cuales el aprendizaje automático tiene o ha tenido un papel relevante.

En la segunda parte de este capítulo, se profundiza en el aprendizaje automático no supervisado y más concretamente en las técnicas de agrupamiento. En esta parte nos preguntamos por la motivación y el origen de estos métodos de agrupamiento, explicando, además, las distintas medidas de similitud o distancia necesarias para su implementación. A continuación, se introducen brevemente los distintos tipos de algoritmos de agrupamiento existentes en la literatura

actual. Para terminar, al igual que en la parte anterior, se resumen las distintas aplicaciones de las técnicas de agrupamiento.

2.1. Aprendizaje automático.

El aprendizaje automático es una rama de la inteligencia artificial originada para desarrollar técnicas de aprendizaje en los ordenadores. El origen del aprendizaje automático es poco exacto, ya que muchas de las técnicas utilizadas son anteriores a la formalización de este concepto. El origen de la inteligencia artificial moderna tuvo lugar en la conferencia llevada a cabo en Dartmouth en 1956. Algunos de sus participantes, J. MacCarthy, M. Misky, A. Newell y H. Simon, se convirtieron en los principales referentes de esta materia. No obstante, la primera vez que se habló del aprendizaje automático como tal fue en la década de los años 50, cuando A. Samuel diseñó el primer programa de ajedrez capaz de competir con un campeón mundial.

Cuando una persona intenta resolver un problema prueba una serie de métodos simples. Primero, recuerda aquellos métodos usados en el pasado que obtuvieron un resultado positivo y descarta aquellos métodos cuyo resultado fuese negativo. Cuando surgen problemas de mayor complejidad, primero prueba a resolverlos usando aquellos métodos que fueron exitosos en el pasado. Si estos métodos no sirven para resolver los nuevos problemas, tratará de combinarlos obteniendo un nuevo conjunto de métodos. Aquellos que obtienen un resultado positivo con mayor frecuencia, serán aquellos usados en el futuro. Éste proceso descrito resume los distintos pasos a seguir por una persona para resolver un problema.

Del mismo modo que actúa un hombre puede actuar una máquina. En general, se dice que *un programa aprende de una experiencia E con respecto a algunos tipos de tareas T y una medida de rendimiento P, sí el rendimiento en las tareas T medido por P, mejora con la experiencia E [5].*

El aprendizaje automático se encarga de diseñar y desarrollar algoritmos que permitan a los ordenadores ser más eficientes y realizar tareas sin apenas supervisión humana. El aprendizaje automático tratará de producir de manera automática modelos, como pueden ser reglas o patrones, de una serie de datos iniciales. El aprendizaje automático está por tanto íntimamente relacionado con campos tan extensos como pueden ser la minería de datos, la estadística o el reconocimiento de patrones, entre otros.

Dada la importancia de este campo y su gran crecimiento en las últimas décadas, han surgido numerosos métodos y técnicas para conseguir que una máquina o un programa aprendan de una experiencia. Cada uno de ellos tiene unas características determinadas y ofrecen al usuario una aplicación específica al problema que se intenta resolver. En el siguiente apartado se hace una taxonomía de los métodos más comunes en el campo del aprendizaje automático y se hace una introducción de aquellos que han sido más significativos a lo largo de la historia.

2.1.1. Métodos de aprendizaje.

El aprendizaje automático tiene como objetivo final el desarrollo de técnicas que permitan a los ordenadores aprender a partir de ejemplos. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información suministrada, y que consigan mejorar los resultados a partir de la experiencia. Este es, por tanto, un proceso de inducción del conocimiento.

Existen dos modos diferenciados en el proceso del aprendizaje automático dependiendo de si se posee información acerca de la salida o no. Estos modos son denominados como supervisado y no supervisado respectivamente. Por otro lado, recientemente han surgido nuevos métodos de aprendizaje que, por su creciente importancia, han pasado a considerarse entidades propias en la clasificación de los métodos de aprendizaje automático. Estos son denominados como aprendizaje por refuerzo, y aprendizaje semi-supervisado. En los siguientes puntos se estudian estos tipos de aprendizaje y se detallan brevemente alguno de los modelos más utilizados en cada uno de ellos.

2.1.1.1. Aprendizaje supervisado.

Con aprendizaje supervisado nos referimos a todas aquellas aplicaciones o procesos en los que se dispone de información tanto de los valores de entrada del sistema como de los valores de salida deseados. De manera global, dos de los problemas típicos en el aprendizaje supervisado son el de clasificación y el de regresión. En clasificación, los valores deseados se corresponden con las etiquetas de cada caso (información cualitativa), mientras que en regresión, la información de salida es el valor real a estimar (información cuantitativa).

Un problema de clasificación es, por ejemplo, el reconocimiento de caracteres, cuyo objetivo final es asignar a cada vector de entrada una de las categorías discretas. Expresado de otro modo, en la clasificación supervisada el mapeo de un conjunto de vectores de entrada ($x \in \mathbb{R}^d$, donde d es la dimensión del espacio de entrada) a un conjunto finito de etiquetas discretas ($y = 1 \dots C$, donde C es el número de clases existentes), donde dicho mapeo es modelado en términos de una función matemática $y = f\,(x, w)$, siendo w un vector de parámetros ajustables. El vector de parámetros se determina mediante un algoritmo de aprendizaje.

Si por el contrario, la salida deseada consiste en una o más de una variable continua, entonces nos encontramos ante un problema de regresión. Un ejemplo de un problema de regresión es, por ejemplo, la predicción del rendimiento en un proceso de manufacturación químico, en el cual las entradas consisten en concentraciones de reactantes, valores de temperatura o valores de presión.

En la figura 2.1 puede verse el esquema de un proceso genérico de aprendizaje supervisado. En este proceso, la primera fase es la de obtención de los datos que son necesarios para definir el modelo. Tras esta primera fase, es muy frecuente encontrarnos con la necesidad de transformar

los datos obtenidos y llevarlos a otro espacio que pueda facilitarnos la creación de un modelo aprendido y con mayor precisión. Esta fase de preproceso suele denominarse fase de extracción y/o selección de características. Se ha de tener en cuenta que el conjunto de datos resultante de estas dos fases debe ser significativo y representativo; es decir, debe haber un número de ejemplos suficientes y diverso, en el cual todas las regiones significativas del espacio están suficientemente representadas para, de este modo, asegurar la aplicación general del modelo.

La siguiente fase es la más importante de todas, y es la denominada fase de aprendizaje. En esta fase se aplican los distintos algoritmos de aprendizaje y finaliza con la obtención del modelo o patrón. Esta fase se puede subdividir en tres subfases: separación de los datos de entrada, selección del algoritmo y entrenamiento. La primera subfase consiste en separar los datos de entrada en dos subconjuntos totalmente independientes: los datos de entrenamiento, que se emplean durante el entrenamiento, y los datos de validación, que se utilizan para la validación posterior del modelo generado. La siguiente subfase es la elección del algoritmo apropiado para el problema planteado, teniendo en cuenta el tipo de datos del conjunto, los parámetros de entrada que son conocidos, etc. Existen un gran número de algoritmos que ofrecen distintas posibilidades, el objetivo de esta fase es el de escoger aquel que mejor se adapte a las necesidades del problema. La última subfase del proceso de aprendizaje es la de mayor relevancia; es la fase de entrenamiento y en ella se ejecuta el algoritmo escogido para los parámetros dados. Durante esta fase el algoritmo se ejecuta de manera iterativa obteniendo un error en cada iteración, siendo este error la diferencia entre el valor esperado y el obtenido por el modelo aprendido. Este modelo se ajustará en las sucesivas iteraciones en función de este error. Al final de esta fase se obtendrá un modelo aprendido ajustado a los datos de entrenamiento.

Por último, se ha de comprobar que el modelo obtenido nos ofrece un resultado idóneo para el problema. Para ello se utiliza un nuevo conjunto de datos, los datos de validación. En esta etapa se comparan los resultados del modelo aprendido con los datos de validación, obteniendo así el error real del modelo. Si el error es inferior a un umbral determinado el proceso de aprendizaje se dará por concluido. Si es mayor, el diseñador del proceso podrá volver a alguna de las fases iniciales para así reajustar o rediseñar el proceso global. Estas fases serán repetidas hasta que el error final sea inferior al umbral definido.

Todas estas fases desempeñan un papel fundamental, y un fallo en alguna de ellas puede llevar a un fallo de clasificación o de generación del modelo. De entre ellas, la fase principal es la de aprendizaje, ya que ésta es la que terminará generando el patrón o modelo.

Son muchas las técnicas que han surgido dentro del aprendizaje supervisado. Una de las primeras técnicas fue K-vecinos más cercanos (KNN *K-Nearest Neighbours*) y posteriormente surgieron otros métodos como las redes neuronales, los árboles de decisión y las máquinas de vectores de soporte. A continuación, se describen brevemente cada una de ellas.

K-Nearest Neighbours. El método K-Nearest Neighbours (KNN, en español: K vecinos más cercanos), definido por Fix y Hodges en 1951, es un método de clasificación supervisada que sirve para estimar la función de densidad $F\left(x|C_j\right)$ o directamente la probabilidad a posteriori de que un elemento x pertenezca a la clase C_j a partir de la información proporcionada por

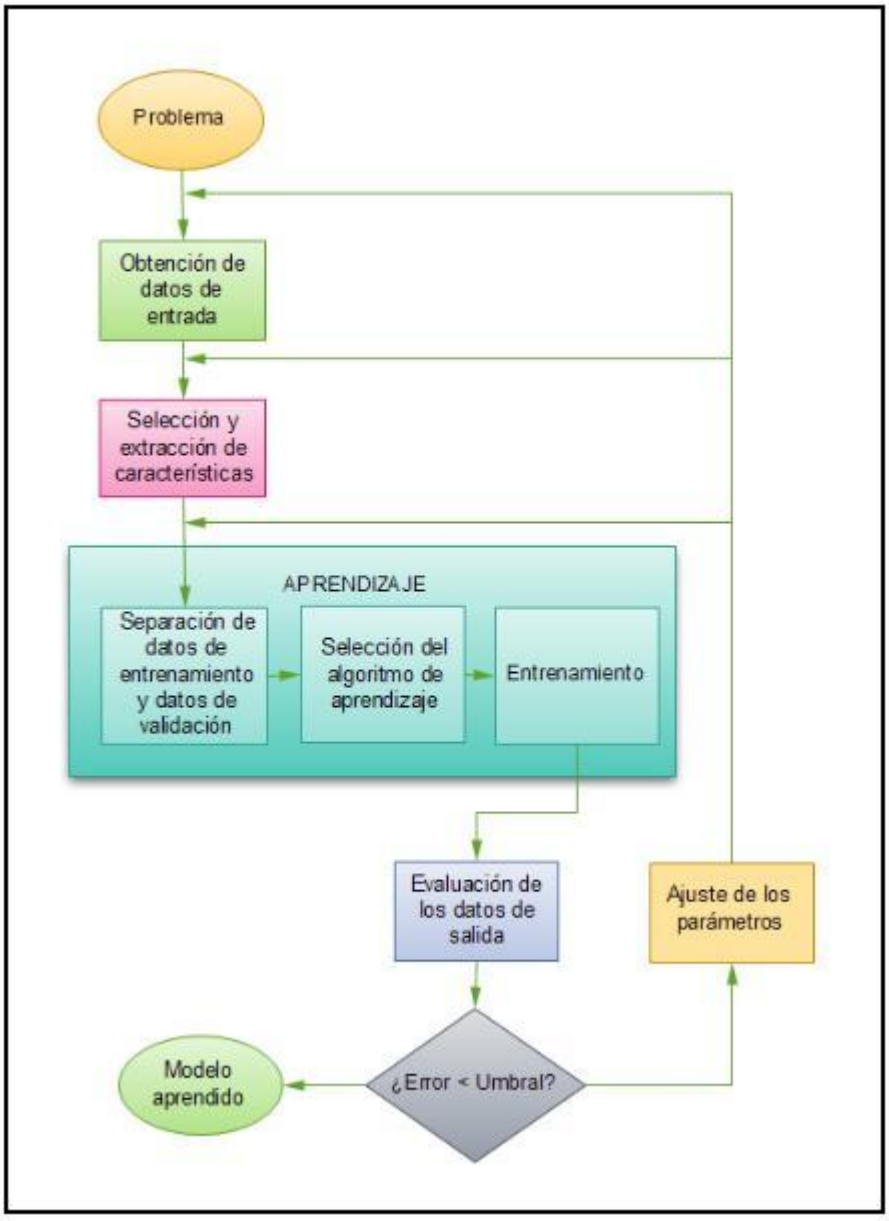


Figura 2.1: Diagrama de flujo del proceso de aprendizaje supervisado.

el conjunto de prototipos. En el proceso de aprendizaje de este método no se hace ninguna suposición acerca de la distribución de las variables predictoras.

El algoritmo KNN es uno de los algoritmos más simples y consiste en asignar un objeto a la clase más común entre sus K vecinos más cercanos, siendo K un número entero positivo. Los vecinos se obtienen de un conjunto de objetos (denominados datos de entrenamiento), para los cuales el modelo de clasificación correcto es conocido. Para identificar a los vecinos, los objetos son representados por vectores de posición en un espacio de características multidimensional. La métrica más utilizada para ello es la distancia euclídea, aunque también es frecuente el uso de otras distancias como, por ejemplo, la distancia de Manhattan o la de Mahalanobis.

El problema principal del algoritmo KNN es su sensibilidad a la estructura local de los datos de entrada y la falta de un algoritmo de selección de un valor óptimo para K. Aunque, también presenta ventajas, como que el coste de aprendizaje es nulo, que no son necesarias suposiciones iniciales, que es bastante robusto frente al ruido, etc. Existen numerosos métodos que complementan el algoritmo KNN, como por ejemplo K-NN con rechazo, K-NN con distancia media, K-NN con distancia mínima, K-NN con distancia ponderada, etc.

Redes neuronales. Las redes de neuronas artificiales son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida. Una red de neuronas se puede ver como sistemas paralelos de computación masiva con un gran número de procesadores simples con muchas interconexiones. Estas redes se componen de unidades llamadas neuronas o nodos. Cada neurona recibe una serie de entradas a través de interconexiones y emite una salida.

Las primeras redes neuronales surgieron a comienzos del siglo XX, pero no fue hasta la década de los 40 cuando empezaron a cobrar fuerza. En 1957 F. Rosenblatt desarrolló el modelo ampliamente conocido como el PERCEPTRON, y dos años más tarde, en 1959, B. Widrow desarrolló un modelo muy similar al anterior, denominado ADELINE. En [6] se puede encontrar un resumen más detallado sobre la historia de las redes neuronales desde comienzos del siglo XX hasta nuestros días. En la actualidad, uno de los modelos más utilizados es el perceptron multicapa.

Este método se denomina redes de neuronas artificiales debido a la estrecha relación existente entre su estructura matemática y su diseño con el de una red de neuronas biológica. Una red de neuronas biológica se compone de tres partes [6]:

- *Los receptores*: se encuentran en las células sensoriales y recogen la información en forma de estímulos, bien del ambiente o bien del interior del organismo.
- *El sistema nervioso*: recibe las información, la elabora y en parte la almacena y la envía a los órganos efectores y a otras zonas del sistema nervioso.

- *Órganos diana o efectores*: reciben la información y la interpretan en forma de acciones motoras, hormonales, etc. Ejemplos de estos órganos son los músculos y las glándulas.

Del mismo modo que las neuronas biológicas están compuestas de tres partes, las redes neuronales artificiales están formadas por tres niveles de capas:

- *Nivel de entrada*: tiene una sola capa con m neuronas de entrada.
- *Nivel oculto*: puede tener una o más capas, cada una de ellas con n neuronas, siendo n un patrón a escoger por el diseñador de la red.
- *Nivel de salida*: tiene una única capa con c neuronas, siendo c el número de salidas deseadas.

Un ejemplo de red neuronal se representa en la figura 2.2 y en ella se pueden apreciar los tres niveles citados, donde en este caso el nivel oculto tiene una única capa. Este ejemplo de red neuronal es una red neuronal tipo perceptron multicapa.

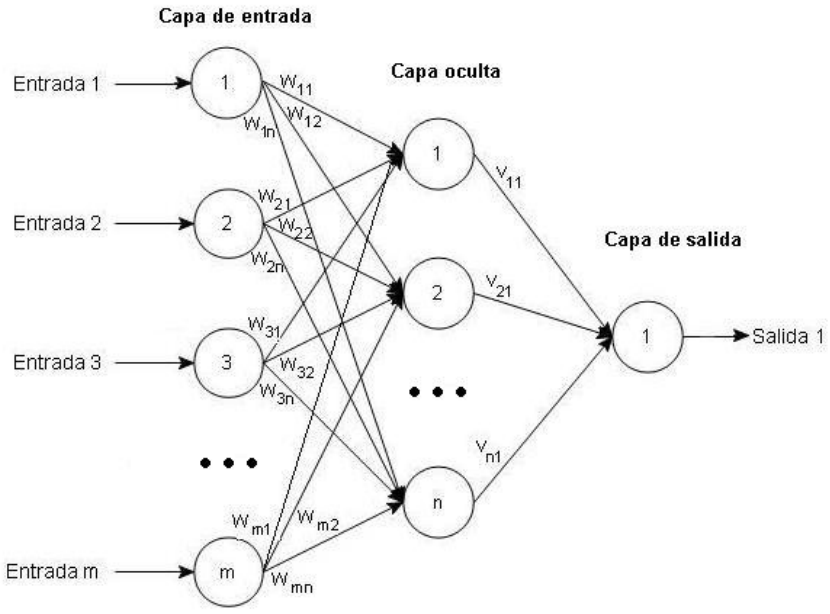


Figura 2.2: Ejemplo de red neuronal tipo perceptron multicapa.

La salida de una red neuronal viene representada por una función $f(x)$ que es la combinación de otras funciones $g_i(x)$, las cuales a su vez pueden ser combinaciones de otras funciones. Una función muy utilizada para una red neuronal con i neuronas ocultas es:

$$f(x) = \sum_i w_i^{(2)} h_i + \theta^{(2)}$$

siendo, w_i los pesos asociados a las neuronas, θ una constante, el superíndice es el nivel correspondiente, y h_i una función no lineal como por ejemplo:

$$h_i = tanh\left(\sum_j w_{ij}^{(1)} x_j + \theta_i^{(1)}\right)$$

La complejidad de esta función y, por tanto, de la red neuronal, dependerá del número de nodos ocultos de la red.

Como se vio en el proceso del aprendizaje automático supervisado, existe una fase de aprendizaje o entrenamiento y otra fase de validación. Durante la fase de entrenamiento, las redes neuronales artificiales usan un conjunto de datos de entrenamiento para determinar los pesos (parámetros de diseño) que definen el modelo neuronal. Una vez entrenado este modelo, se pasa a la fase de validación, en la que se procesan el conjunto de datos de validación, analizándose de esta manera las prestaciones definitivas de la red.

Existen múltiples tipos de redes neuronales que pueden ser clasificados según el tipo de entradas, según los algoritmos de aprendizaje,... Lo más apropiado es escoger la red neuronal teniendo en cuenta las necesidades específicas del problema. La familia de redes neuronales más utilizada para problemas de clasificación es la red de tipo *feed-forward*, la cual incluye redes de perceptron multicapa y redes de funciones de base radiales.

Las redes neuronales adolecen de un problema común en los métodos de aprendizaje supervisado denominado sobre-aprendizaje o sobre-ajuste (*overfitting*). El sobre-aprendizaje ocurre cuando una red neuronal ha aprendido el modelo correctamente en la etapa de entrenamiento, pero no responde adecuadamente a la validación. Esto sucede por diversas causas como, por ejemplo, porque el número de ciclos de aprendizaje es muy elevado, o porque el número de neuronas de la capa oculta es también muy elevado, etc. El mejor modo de evitar el sobre-aprendizaje es lograr un mayor número de casos para el entrenamiento. Cuando esto no sea posible también se puede limitar el número de nodos en la capa oculta, limitar el número de ciclos de aprendizaje o usar la técnica de validación cruzada (*cross-validation*). La figura 2.3 muestra un caso de red neuronal sobre-entrenada. En la figura, el error de entrenamiento se muestra en azul, mientras que el error de validación se muestra en rojo. Si el error de validación aumenta mientras que el de entrenamiento decrece puede que se esté produciendo una situación de sobre-ajuste.

En [7] Bishop hace una revisión más detallada sobre las redes neuronales, sus tipos, sus aplicaciones, etc.

Árboles de decisión. Un árbol de decisión es una herramienta de soporte que usa grafos o modelos de decisión, incluyendo todas sus propiedades: costes, probabilidades de que ocurra un evento, utilidad, etc.

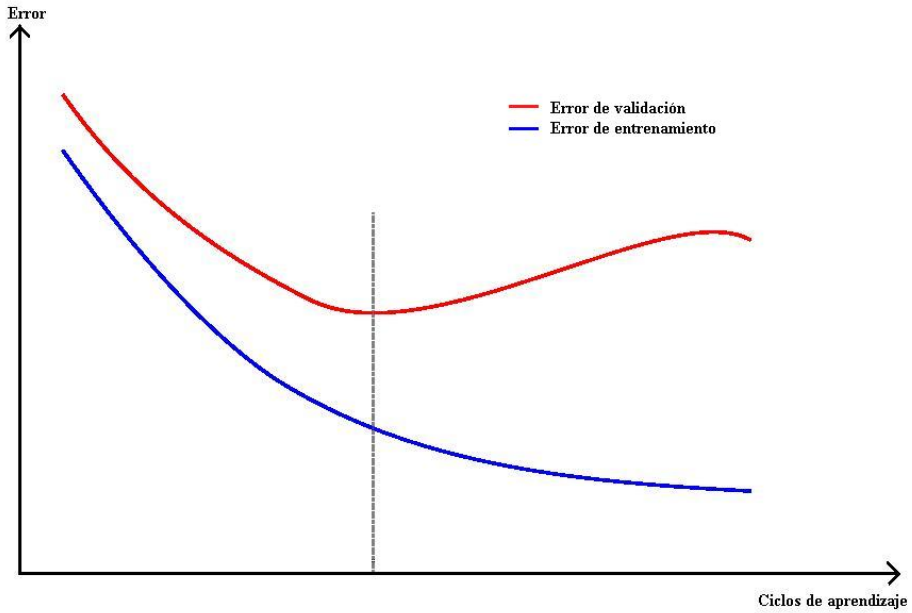


Figura 2.3: Sobre-ajuste o sobre-aprendizaje (*overfitting*) en una red neuronal.

El uso de árboles de decisión tuvo su origen en las ciencias sociales con los trabajos de Sonquist y Morgan (1964), y Morgan y Messenger (1979) realizado en el *Survey Research Center* del *Institute for Social Research* de la Universidad de Michigan. Pero, en el campo del aprendizaje automático, no fue hasta 1984 cuando Breiman, Friedman, Olshen y Stone [8] introdujeron el algoritmo conocido como CART (*Classification And Regression Trees*) para la construcción de árboles y se aplicó a problemas de regresión y clasificación. Casi al mismo tiempo, el proceso de inducción mediante árboles de decisión comenzó a ser usado por la comunidad de aprendizaje automático (Michalski, (1973), Quinlan (1983)). Uno de los árboles de decisión más conocidos, el C4.5, fue introducido por Quinlan en 1993 y desciende del ID3, también creado por Quinlan en 1983 [9].

Los árboles de decisión se emplean con frecuencia en operaciones de búsquedas, especialmente en análisis de decisión, para ayudar a identificar la estrategia con mayor probabilidad de alcanzar un objetivo. Otro uso de los árboles de decisión es el de un medio descriptivo para calcular probabilidades condicionales.

La clasificación de un patrón particular en árboles de decisión comienza en el nodo raíz, el cual pregunta por el valor de una propiedad particular del patrón. Los diferentes enlaces que salen del nodo raíz se corresponden con los posibles valores que puede tomar. Dependiendo de este valor, descenderemos a través del enlace hasta un nodo hijo. En los árboles, los enlaces han de ser mutuamente distintos y exhaustivos, es decir, sólo se puede seguir a través de un único enlace. El siguiente paso es tomar una decisión en el sub-árbol del nodo hijo, que será tomado como nodo raíz del sub-árbol. Seguiremos así hasta que lleguemos a un nodo hoja, esto es, un

nodo que no tiene hijos. Cada nodo hoja contiene una etiqueta con la categoría correspondiente, y el patrón de prueba es asignado con la categoría del nodo hoja alcanzado.

El árbol de decisión simple de la figura 2.4 ilustra el principal beneficio de esta técnica de clasificación frente a otras: la facilidad de interpretación. Pero, por otro lado, su principal inconveniente es la dificultad de implementación para conjuntos de datos de gran dimensionalidad.

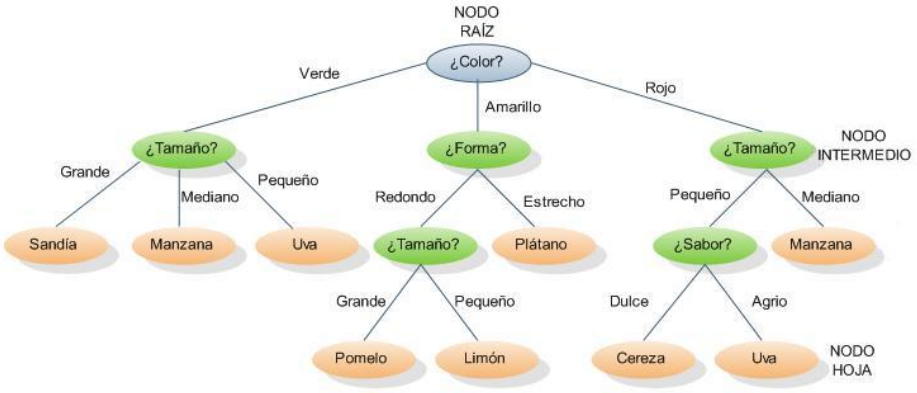


Figura 2.4: Ejemplo de un árbol de decisión.

S.R. Safavian y D. Landgrebe presentaron en [10] un estudio de la clasificación a través de árboles de decisión, y en [11, 12, 13, 14] también encontramos diversos estudios interesantes sobre árboles de decisión.

Máquinas de Vectores de Soporte. Las Máquinas de Vectores de Soporte (*Support Vector Machine*, por sus siglas en inglés SVM) son un nuevo sistema de aprendizaje que ha tenido un desarrollo muy significativo en los últimos años, tanto en la generación de nuevos algoritmos como en las estrategias para su implementación. SVM es un sistema de aprendizaje basado en el uso de un espacio de hipótesis de funciones lineales en un espacio de mayor dimensión inducido por un Kernel. En este nuevo espacio las hipótesis son entrenadas por un algoritmo tomado de la teoría de optimización, que utiliza elementos de la teoría de generalización. SVM es un sistema para entrenar máquinas de aprendizaje lineal de manera eficiente. Tanto para clasificación como para regresión se han encontrado muchas aplicaciones de SVM, como por ejemplo en clasificación de imágenes, en reconocimiento de caracteres, en detección de proteínas, en clasificación de patrones, en identificación de funciones, etc.

Las máquinas de vectores de soporte pueden verse como una solución a los problemas de las redes neuronales. Las máquinas de vectores de soporte se caracterizan por ser problemas de optimización convexos. Uno empieza formulando un problema en un espacio inicial determinado, pero termina resolviendo el problema en otro espacio de mayor dimensión donde existe una solución lineal. La solución obtenida por las máquinas de vectores de soporte son soluciones dispersas.

SVM esta basado en el principio de minimización del riesgo estructural, el cual ha demostrado ser superior al principio de minimización del riesgo empírico, utilizado por las redes neuronales convencionales. Algunas de las razones por las que este método ha tenido éxito es que no padece de mínimos locales y el modelo sólo depende de los datos con más información llamados vectores de soporte (SV por sus siglas en ingles, «Support Vectors»).

La máquina de vectores de soporte más sencilla fue la ideada por Vapnik (para información más detallada de esta técnica ver [15]). Una visión general de las máquinas de vectores es la ofrecida por C.J.C. Burges en [16], aunque existe una bibliografía muy extensa sobre este tema.

2.1.1.2. Aprendizaje no supervisado.

Hasta ahora, hemos hablado del aprendizaje supervisado donde se dispone de información de la salida. En el lado contrario nos encontramos con los problemas en los que no se dispone de ningún tipo de información de salida sobre los datos, pero se desea organizar los datos de alguna manera para mejorar su comprensión. Este tipo de problemas son los denominados problemas de aprendizaje no supervisado.

Según [14], existen cinco razones principales por las cuales el uso de técnicas de aprendizaje no supervisado resulta de interés para las aplicaciones:

- La recolección de datos y su posterior etiquetamiento en un conjunto de datos muy extenso puede suponer un coste muy elevado.
- También puede ser de interés actuar en sentido contrario; aprender con una gran cantidad de datos sin etiquetar, y sólo usar supervisión para etiquetar los distintos grupos encontrados.
- En muchas aplicaciones las características de los patrones cambian lentamente con el tiempo. Si estos cambios pueden rastrearse en un proceso de ejecución sin supervisar, podremos obtener un resultado más idóneo.
- Podemos usar métodos no supervisados para encontrar características que serán útiles para la categorización.
- En el comienzo de la investigación puede ser útil tener una visión general de la naturaleza y la estructura de los datos.

En la figura 2.5 puede verse el esquema de un proceso genérico de aprendizaje no supervisado. Este proceso es similar al proceso de aprendizaje supervisado. Al igual que en el anterior, existen varias fases fundamentales. Las primeras fases, la de obtención de datos y su preproceso (selección y extracción de características), son idénticas a las fases del aprendizaje supervisado. En cambio, la siguiente fase, el aprendizaje, no es la misma. En este caso, al no disponer de información acerca de la salida, en la fase de entrenamiento no se puede reajustar el modelo en

base al error. Pero sigue siendo necesario separar los datos en datos de entrenamiento y datos de validación para decidir si el método es bueno o no. La fase de selección del algoritmo y el entrenamiento también se mantienen. En este caso, la posibilidad de validar si los resultados son correctos no es frecuente, puesto que no se dispone de información de la salida. La manera de decidir cuando se ha aprendido es viendo si el sistema converge o estableciendo un criterio de parada como puede ser un número de iteraciones de funcionamiento máximo.

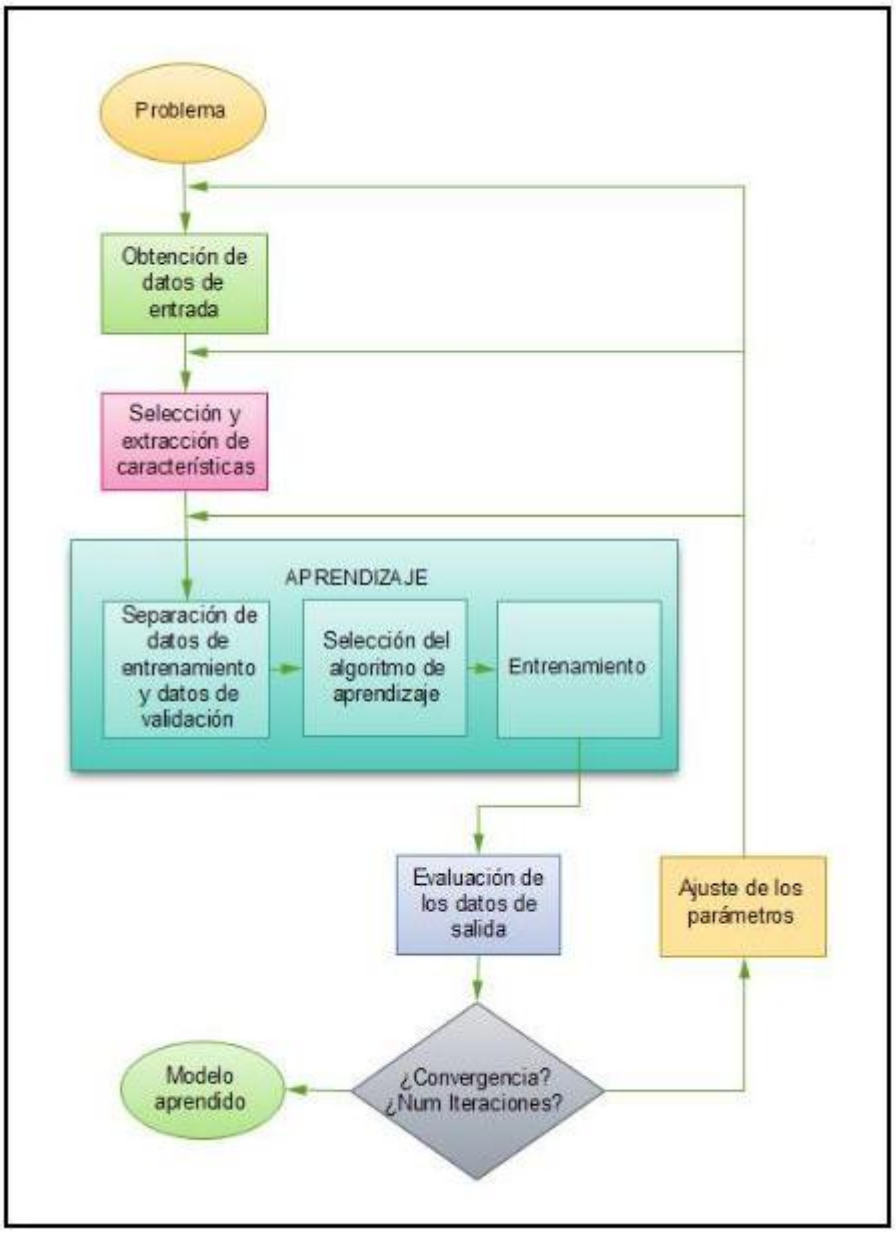


Figura 2.5: Diagrama de flujo de un proceso de aprendizaje no supervisado.

Existen dos tipos de técnicas fundamentales de aprendizaje no supervisado: agrupamiento o *clustering* y redes neuronales no supervisadas.

Agrupamiento o *Clustering*. El agrupamiento se puede considerar como la aproximación más utilizada en aprendizaje no supervisado. Su objetivo general es encontrar algún tipo de estructura en una colección de datos sin etiquetar o sin clasificar, ya que en la mayoría de los casos no se dispone de este tipo de información.

Los algoritmos de agrupamiento buscan organizar objetos en distintos grupos cuyos miembros tienen características similares. Un *cluster* o grupo es por tanto una colección de objetos que son similares entre ellos y diferentes respecto a los miembros de otros grupos.

En el apartado 2.2, se hará una descripción más detallada de las técnicas de agrupación y sus tipos. Posteriormente se hará hincapié en las técnicas de agrupamiento basadas en densidades, algunas de las cuales serán objeto de estudio y comparación.

Redes neuronales no supervisadas. Las redes neuronales explicadas hasta ahora eran supervisadas. Esto es, necesitan obtener información acerca de la salida para poder realizar su aprendizaje. Sin embargo, en muchas situaciones no es posible tener esta información a priori. En este apartado, se estudian un nuevo tipo de redes neuronales no supervisadas, en la que las propias redes son capaces de modificar sus parámetros internamente sin necesidad de supervisión.

Las redes neuronales no supervisadas, por lo general, tienen una arquitectura sencilla, y se caracterizan por ser más similares a los modelos biológicos que las redes neuronales artificiales supervisadas.

La primera aproximación a las redes neuronales no supervisadas fue en el año 1949, cuando D. Hebb enunció la regla que lleva su mismo nombre, y que se refiere al comportamiento biológico observado en las neuronas. Más tarde, en 1987, Carpenter y Grossberg desarrollaron los modelos ART de redes neuronales no supervisadas basándose en la teoría de resonancia adaptiva.

Uno de los modelos de redes neuronales no supervisadas más usados es el desarrollado por Kohonen basado en mapas auto-organizativos (*self-organizing maps*). Las redes de Kohonen son redes de dos capas: la capa de entrada que recibe la señal de entrada a la red, y la capa de salida que realiza el cálculo. Cada neurona de la capa de salida debe reflejar las coordenadas que tiene en el espacio que el diseñador de la red decida. Para que las neuronas puedan ser comparadas con la posición de otras neuronas de la red, a cada neurona se le asocia una regla de vecindad. El objetivo es agrupar entradas similares a neuronas de posiciones similares. El conjunto de datos es agrupado, porque es asociado al mismo nodo o al mismo conjunto de nodos.

Existe una amplia bibliografía acerca de este tipo de redes. Ejemplos de aplicaciones de redes neuronales no supervisadas y un estudio más profundo sobre éstas puede encontrarse en [6].

2.1.1.3. Aprendizaje por refuerzo.

Otra técnica muy extendida dentro del aprendizaje automático es el aprendizaje por refuerzo. Este tipo de aprendizaje busca las acciones más apropiadas dada una situación concreta, de manera que aumente la recompensa por actuar de ese modo. El objetivo del aprendizaje por refuerzo es usar el paradigma *premio-castigo* para aprender una función, la cual permitirá tomar decisiones en el futuro a partir de la percepción del entorno. Es frecuente encontrar una secuencia de acciones y de estados por medio de los cuales el algoritmo interactúa con el entorno. En muchos casos, la acción actual no sólo afecta al beneficio o recompensa inmediatos, sino que también puede afectar al beneficio en otros momentos futuros.

Una de las características del aprendizaje por refuerzo es que se puede modelar con una cadena de Markov. En estas cadenas, la acción a escoger dada una situación depende únicamente de esta situación, y no del camino seguido para llegar a ella.

Las aplicaciones del Aprendizaje por Refuerzo son múltiples, desde robots móviles que aprenden a salir de un laberinto, programas de ajedrez que aprenden cuáles son las mejores secuencias de movimientos para ganar un juego, o un brazo robótico que aprende cómo mover las articulaciones para lograr el movimiento final deseado.

2.1.1.4. Aprendizaje semi-supervisado.

Hasta ahora, hemos visto que el aprendizaje se puede realizar teniendo datos acerca de la salida o sin ellos. El aprendizaje semi-supervisado es una combinación del aprendizaje supervisado y no supervisado. Puesto que asignar etiquetas o clases a los datos puede ser muy costoso, se puede recurrir a la opción de usar a la vez un conjunto de datos etiquetados de tamaño pequeño y un conjunto más extenso de datos no etiquetados, mejorando así la construcción de los modelos. Esta técnica es la usada por el aprendizaje semi-supervisado. En este método, se ha de tener en cuenta que no siempre los datos no etiquetados son de ayuda al proceso de aprendizaje. Por lo general, se asume que los datos no etiquetados siguen la misma distribución que los etiquetados para que el uso de datos sin etiquetar sea útil.

Existen numerosos métodos de aprendizaje semi-supervisado, los cuales ofrecen unas características determinadas. La manera de escoger el método más adecuado es viendo cual de ellos se ajusta mejor a las necesidades del problema específico.

Uno de estos métodos de aprendizaje es el co-entrenamiento (*co-training*) [17, 18], el cual asume que el espacio de características se puede dividir en dos subconjuntos independientes de atributos. Cada subconjunto de atributos es suficiente para aprender un clasificador adecuado. Inicialmente, dos clasificadores separados son entrenados con los datos etiquetados de los dos subconjuntos respectivamente. Posteriormente, cada clasificador clasifica los datos sin etiquetar, y «enseña» al otro clasificador con algunos de los datos sin etiquetar de mayor fiabilidad o confianza. Cada clasificador es entonces re-entrenado con los nuevos ejemplos entregados por

el otro clasificador. El proceso se repite hasta que los dos clasificadores estén de acuerdo, tanto en los datos sin etiquetar, como en los etiquetados.

Otro método de aprendizaje semi-supervisado es el de componentes comunes usando *Expectation-Maximization*. La idea básica de este método es entrenar un clasificador usando únicamente ejemplos etiquetados, y usar ese clasificador para asignar probabilidades-pesos de etiquetas a los ejemplos no etiquetados. Después se entrena un nuevo clasificador usando esas estimaciones y se vuelven a asignar pesos. Aunque los datos no etiquetados por sí solos no nos dan más información que una aleatoria si la clase es desconocida, la información de los valores de los atributos nos proporciona información útil sobre su distribución.

Se puede encontrar información más detallada sobre estos y otros métodos de aprendizaje semi-supervisado en [19].

2.1.2. Aplicaciones del aprendizaje automático.

Hasta ahora se ha estudiado el significado del aprendizaje automático y se han visto algunas técnicas del aprendizaje automático. El objetivo final del aprendizaje automático es poder realizar nuevas aplicaciones que ayuden y complementen las aplicaciones hasta ahora existentes.

El aprendizaje automático es una rama descendiente de la estadística y de la informática. Por tanto, son numerosas las aplicaciones del aprendizaje automático en estas áreas. En el área de la informática, el aprendizaje automático está relacionado con aplicaciones tan diversas como el desarrollo de robots humanoides o Internet. Un ejemplo de estas aplicaciones es el *PageRank* usado por *Google*, que ampara una familia de algoritmos utilizados para asignar de forma numérica la relevancia de los documentos (o páginas web) indexados por un motor de búsqueda [20]. Por otro lado, en el campo de la estadística, el aprendizaje automático busca obtener conclusiones en el análisis de los conjuntos de datos.

Estos métodos de aprendizaje automático no sólo se utilizan en el campo de la estadística y en la informática para desarrollar nuevas aplicaciones. El área de la psicología, la neurociencia y áreas similares también aplican estas técnicas. El objetivo de estas aplicaciones es en este caso estudiar el aprendizaje y las conductas en humanos y animales.

La biología y la medicina son también áreas de aplicación del aprendizaje automático. El aprendizaje automático es muy importante para la clasificación de secuencias de ADN o en el estudio de enfermedades y su posible predicción.

Por último, el aprendizaje automático también puede ser aplicado en los campos de la economía o del control de sistemas, con un objetivo común: la posibilidad de adaptarse o optimizar automáticamente el entorno. Así, una posible aplicación en la economía es el estudio de los mercados, la búsqueda de clientes o la detección de fraudes. En sistemas de control, el aprendizaje automático puede ser aplicado en sistemas de servo-control, los cuales pueden mejorar su estrategia de control a través de la experiencia.

En la tabla 2.1 figuran algunas de las aplicaciones del aprendizaje automático en estos campos.

Dominio del problema	Aplicación	Información
Bioinformática	Análisis de secuencias	ADN / Secuencias de proteínas.
Medicina	Diagnóstico médico	Estudios de enfermedades, diagnósticos previos.
Economía	Detección de fraudes	Comportamientos de clientes o informes de facturas.
Economía	Predicción de la bolsa	Documentos e informes bursátiles.
Reconocimiento biométrico	Identificación de personas	Cara, iris, huellas dactilares.
Reconocimiento de voz	Información de directorios telefónicos sin necesidad de operadores	Onda de voz.
Astronomía	Clasificación morfológica de galaxias	Imágenes del espacio
Clasificación de documentos	Búsqueda en Internet	Documentos de texto
Análisis de imágenes	Lectura automática para ciegos	Documentos de imagen
Robótica	Reproducir el comportamiento humano	Sensores de audio, video, infrarrojos, etc.
Teoría de juegos	Automatización de juegos	Estrategias
Psicología y Robótica	Reconocimiento de emociones humanas	Análisis de ondas de voz y expresiones faciales
Informática	PageRank	Páginas web e información de accesos

Tabla 2.1: Ejemplos de aplicaciones del aprendizaje automático y sus dominios.

2.2. Métodos de agrupamiento o *clustering*.

Durante toda su historia, el hombre ha obtenido conocimiento a partir de la clasificación de objetos. El filósofo Platón fue consciente de este hecho, y al hablar sobre el mundo de las ideas consiguió representar cómo funciona la inteligencia humana. Platón asentó las bases de lo que hoy en día entendemos por conocimiento. La mente funciona clasificando objetos. La capacidad de reconocer un objeto es la capacidad de obtener un concepto que se identifique con este objeto y de poder clasificar todos aquellos objetos que representan a un mismo concepto. De esta manera, aunque existan infinidad de mesas distintas (hay mesas grandes, pequeñas, redondas, cuadradas, de roble, de pino,...) todas presentan algo en común, todas tienen un tablero y unas patas que lo sostienen. Este concepto, este modelo ideal, es lo que nos permite identificar una mesa cuando la vemos.

La sociedad actual se caracteriza por la cantidad de información a su alcance. El uso de ordenadores y el incremento de la capacidad de almacenaje permiten que cada vez se guarden más datos. Toda esta información es procesada y analizada para obtener de ella alguna información relevante, que pueda ser manejada para uso propio o para futuros análisis. Para poder comprender un objeto, las personas tienden a caracterizarlo y a compararlo con otros objetos existentes, basándose en su similitud o su disimilitud. Para poder caracterizarlo y compararlo es necesario disponer de la máxima cantidad de información.

Uno de los métodos de clasificación de datos es el agrupamiento. Este consiste en ordenar observaciones o vectores de características en grupos (*clusters*), sin tener ningún tipo de información sobre la salida, esto es, sin disponer de datos etiquetados. Al no poseer ningún conocimiento acerca de los patrones de salida, nuestro sistema de clasificación tiene que descubrir la estructura interna de similitud de los datos de forma eficiente. El agrupamiento es útil en muchas técnicas exploratorias de análisis de patrones, minería de datos, toma de decisiones,... es decir, es útil en muchas técnicas de aprendizaje automático. Sin embargo, en muchos de estos problemas existe poca información a priori (como sucede por el contrario en los modelos estadísticos) y a la hora de tomar las decisiones se deben hacer el mínimo número posible de asunciones sobre los datos.

Podemos definir como agrupamiento al proceso de agrupar objetos, de tal modo que los objetos de un mismo grupo son más similares unos a otros que con objetos de otros grupos. Siguiendo la definición en [11] este concepto se puede formalizar como: dado un conjunto de datos de entrada $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$, tal que $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]^T$, siendo cada componente $x_{ij} \in \mathbb{R}^d$ una característica distinta, se define una «m-agrupación» de X a una partición del conjunto de datos X en m conjuntos o *clusters*, $C_1, C_2, \ldots, C_m$, de tal manera que se cumplan estas tres condiciones:

- $C_i \neq 0, i = 1, \ldots, m$
- $\cup_{i=1}^{m} C_i = X$
- $C_i \cap C_j = 0, \forall_{i,j} : i \neq j, i, j = 1, \ldots, m$

El conjunto de vectores contenidos en C_i son «más similares» entre ellos y «menos similares» entre los contenidos en C_j.

Según esta definición dada, a cada objeto o punto se le asigna una única clase, grupo o *cluster*, es decir, la agrupación llevada a cabo es de tipo dura. Pero también podría darse el caso de que un vector pudiera pertenecer a varias clases con un cierto grado de pertenencia, es el denominado *fuzzy clustering* [21]. Un agrupamiento tipo *fuzzy* del conjunto de datos X en m grupos, se caracteriza por m funciones u_j, denominadas funciones de pertenencia, donde $u_j : X \rightarrow [0, 1], j = 1, \ldots, m$ y $\sum_{j=1}^{m} u_j(\mathbf{x}_i) = 1, i = 1, \ldots, N$ y teniendo en cuenta que $0 \leq \sum_{i=1}^{N} u_j(\mathbf{x}_i) \leq N, j = 1, \ldots, m$. El valor de estas funciones de pertenencia es una caracterización matemática de un conjunto, en nuestro caso, el grupo. Aquellos valores más cercanos a la unidad representan un mayor grado de pertenencia a un grupo, y aquellos cercanos al cero

representan un menor grado de pertenencia. Los valores de estas funciones de pertenencia son un indicativo de la estructura del conjunto de datos, en el sentido de que si una función de pertenencia tiene valores cercanos a la unidad para dos vectores del conjunto X, por ejemplo x_n y x_k, estos se pueden considerar similares entre ellos.

En definitiva, se puede afirmar que el objetivo final de las técnicas de agrupamiento es organizar, a partir de medidas de similitud o disimilitud, los datos en grupos o *clusters* con algún significado. Estos grupos tienen la particularidad de ofrecernos algún tipo de utilidad de la que antes no disponíamos. No hay que olvidar, que para la agrupación los datos iniciales no contienen ningún tipo de dato etiquetado, es decir, se trata de un método de clasificación no supervisado.

Cuando se desea aplicar alguna de las técnicas de agrupamiento a un problema concreto, se deben tener en cuenta una serie de fases para obtener un resultado óptimo. Estas etapas pueden variar según las necesidades y las particularidades de los datos, pero, en general, según se explica en [11], estas fases se pueden resumir en:

- Preproceso: en algunos casos es necesario un tratamiento previo de los datos. Esto puede deberse, por ejemplo, a su alta dimensionalidad. Ésta se trata de una fase especialmente importante ya que su inclusión puede originar resultados finales totalmente distintos. En ella se transforma el espacio original de los vectores de entrada en un nuevo espacio de variables, donde el problema pueda ser resuelto con mayor facilidad. Existen dos tipos de preprocesado:
  - Extracción de características. La extracción de características se encarga de extraer aquellos datos que son más relevantes para la clasificación y reducir la dimensionalidad de los datos de entrada.
  - Selección de características. La selección de características consiste en escoger un subconjunto de datos de entrada eliminando aquellas características con poca o ninguna información predictiva. Más información sobre esta fase puede obtenerse en [22].
- Diseño del algoritmo. Esta fase es la principal y se puede dividir en tres fases:
  - Medida de similitud o distancia. Para saber cuánto se asemejan o cuánto difieren dos vectores de características o dos objetos, es necesario definir una métrica de similitud o distancia. Existen distintas medidas de similitud, y escoger la adecuada es una tarea a la hay que prestar atención, ya que esta medida afecta directamente a la formación de los grupos o *clusters* resultantes.
  - Criterio para el agrupamiento. Dependiendo del tipo de grupos que se quiera encontrar, de si los parámetros de entrada son conocidos, etc. se deberán tener en cuenta unos criterios u otros para agrupar los datos.
  - Selección del algoritmo. Una vez definidas la medida de similitud y los criterios de agrupamiento, podemos pasar a escoger un algoritmo que se adapte a nuestros requisitos. Existen muchos tipos de algoritmos que ofrecen distintas soluciones a diversos problemas, por ello las fases anteriores son muy importantes.

- Validación de los resultados. Una vez se obtengan los resultados, es necesario verificar que son correctos y que no hay fallos en la clasificación. El proceso de agrupamiento va a dar siempre una salida, pero esta puede ser correcta o no. Por ello, es importante validar que el resultado obtenido es el esperado.
- Interpretación de los resultados. El objetivo final de las técnicas de agrupamiento es obtener algún significado o alguna estructura que nos ofrezca algún tipo de información que antes no poseíamos, o nos ayude a obtener un modelo para otros casos. En esta fase se obtiene esa información o modelo.

En la figura 2.6 se resumen las etapas en el proceso de agrupamiento o *clustering* anteriormente explicadas.

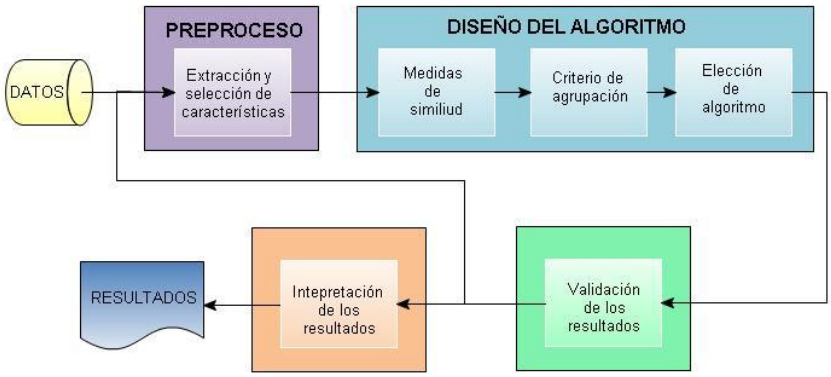


Figura 2.6: Etapas del proceso de agrupamiento.

2.2.1. Medidas de distancia o similitud.

Las medidas de similitud o de distancia son expresiones matemáticas que permiten resumir en un número el grado de relación entre dos entidades, sobre la base de semejanza o la desigualdad entre la cualidad o la cantidad de sus atributos, o ambas. Estas medidas son fundamentales en la definición de agrupamiento y han sido estudiadas desde la década de los 50 [23], en campos tan variados como la psicología o el tratamiento de imágenes [24].

Según [11] una medida de similitud , s, en un conjunto de datos X, se define como $s : X \times X \rightarrow \mathbb{R}$, tal que $\exists s_0 \in \mathbb{R} : -\infty < s(\mathbf{x}, \mathbf{y}) \leq s_0 < +\infty, \forall \mathbf{x}, \mathbf{y} \in X$ y se cumplen:

$$
\begin{aligned}
s(\mathbf{x}, \mathbf{x}) &= s_0, \forall \mathbf{x} \in X \\
s(\mathbf{x}, \mathbf{y}) &= s(\mathbf{y}, \mathbf{x}), \forall \mathbf{x}, \mathbf{y} \in X \\
s(\mathbf{x}, \mathbf{y}) &= s_0 \Longleftrightarrow \mathbf{x} = \mathbf{y} \\
s(\mathbf{x}, \mathbf{y})\, s(\mathbf{y}, \mathbf{z}) &\leq [s(\mathbf{x}, \mathbf{y}) + s(\mathbf{y}, \mathbf{z})]\, s(\mathbf{x}, \mathbf{z}), \forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in X
\end{aligned}
$$

Las medidas de similitud más comunes son:

- *El producto interno*. Se define según la ecuación 2.1. En la mayoría de los casos los vectores **x** e **y** están normalizados, de manera que tengan una misma longitud, a.

$$s_{\text{interno}}(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y} = \sum_{i=1}^{l} x_i y_i \tag{2.1}$$

- *La medida de similitud del coseno*. Se define según la ecuación 2.2 y esta muy relacionada con el producto interno. Posee como ventajas la independencia de la longitud del vector y la invarianza a rotaciones lineales, pero, en cambio, no es invariante a transformaciones lineales.

$$s_{\cos}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\| \, \|\mathbf{y}\|} \tag{2.2}$$

- *El coeficiente de correlación de Pearson*. Esta medida se puede expresar como:

$$r_{\text{Pearson}}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}_d^T \mathbf{y}_d}{\|\mathbf{x}_d\| \, \|\mathbf{y}_d\|}$$

  donde $\mathbf{x}_d$ e $\mathbf{y}_d$ son los vectores diferencia, definidos como: $\mathbf{x}_d = [x_1 - \bar{x}, \ldots, x_l - \bar{x}]^T$ y $\mathbf{y}_d = [y_1 - \bar{y}, \ldots, y_l - \bar{y}]^T$, siendo x_i, y_i la coordenada i de los vectores **x** e **y** respectivamente, y $\bar{x} = \frac{1}{l} \sum_{i=1}^{l} x_i$, $\bar{y} = \frac{1}{l} \sum_{i=1}^{l} y_i$. La ventaja del coeficiente de Pearson sobre el producto interno, es que el primero no depende directamente de los vectores **x** e **y** sino de sus correspondientes vectores diferencia.

La tabla 2.2 muestra un resumen de estas medidas de similitud.

Medida de similitud	Ecuación
Producto Interno	$s_{\text{in}}(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y} = \sum_{i=1}^{l} x_i y_i$
Medida de similitud del coseno	$s_{\cos}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$
Coeficiente de correlación de Pearson	$r_{\text{Pearson}}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}_d^T \mathbf{y}_d}{\|\mathbf{x}_d\| \|\mathbf{y}_d\|}$

Tabla 2.2: Medidas de similitud.

No obstante, más común que medir la similitud entre dos puntos o dos objetos, es medir la disimilitud o la distancia entre ellos. De acuerdo a [11] podemos definir las medidas de disimilitud o la distancia como: $d : X \times X \to \mathbb{R}$, tal que $\exists d_0 \in \mathbb{R} : -\infty < d_0 \leq d(\mathbf{x}, \mathbf{y}) < +\infty, \forall \mathbf{x}, \mathbf{y} \in X$ siendo $\mathbb{R}$ el espacio de los números reales, y cumpliéndose:

$$\begin{aligned} &d(\mathbf{x}, \mathbf{x}) = d_0, \forall \mathbf{x} \in X \\ &d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x}), \forall \mathbf{x}, \mathbf{y} \in X \\ &d(\mathbf{x}, \mathbf{y}) = d_0 \iff \mathbf{x} = \mathbf{y} \\ &d(\mathbf{x}, \mathbf{z}) < d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}), \forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in X \end{aligned} \tag{2.3}$$

Donde d, es una métrica y la inecuación 2.3 es la denominada *desigualdad triangular*. La ecuación anterior a la desigualdad triangular demuestra que el valor d_0 es el mínimo nivel de disimilitud entre dos vectores pertenecientes a X, el cual se consigue cuando son iguales.

Las medidas más comunes de disimilitud o distancia son:

- *La distancia de Minkowski*. Se define según la ecuación 2.4. Posee la peculiaridad de que aquellas características con valores y varianzas grandes tienden a dominar sobre otras características.

$$d_{\text{Mink}}(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{l} |x_i - y_i|^n \right)^{1/n} \tag{2.4}$$

- *La distancia Euclídea*. Es la medida más utilizada. Es un caso especial de la distancia de Minkowski cuando $n = 2$. Es invariante a translaciones y rotaciones lineales. Se define según la ecuación 2.5 y presenta el problema de que tiende a formar grupos esféricos.

$$d_{\text{Eu}}(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{l} |x_i - y_i|^2 \right)^{1/2} \tag{2.5}$$

- *Distancia de Ciudad o de Manhattan*. Al igual que la anterior, es un caso especial de la distancia de Minkowski, pero en este caso $n = 1$. Cuando esta métrica es utilizada para agrupar datos se tiende a formar grupos hiperesféricos. Se define según:

$$d_{\text{Man}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{l} |x_i - y_i|$$

- *Distancia de Pearson*. Se puede considerar como una modificación de la distancia euclídea. La ecuación 2.6 define esta distancia, donde $S_0 = diag\left(s_1^2, \ldots, s_l^2\right)$ es la matriz diagonal que contiene las varianzas de los vectores de X. Esta expresión equivale a reescalar cada variable en unidades de desviación típica. La ventaja de esta medida frente a las anteriores es que es invariante frente a cambios de escala.

$$d_{\text{Pear}}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{l} \frac{(x_i - y_i)^2}{s_i^2}} = \sqrt{(\mathbf{x} - \mathbf{y})^T S_0^{-1} (\mathbf{x} - \mathbf{y})} \tag{2.6}$$

- *Distancia de Mahalanobis*. Se define según 2.7, donde S es la matriz de covarianzas del conjunto de datos X. Es una métrica adecuada como medida de discrepancia de datos debido a los siguientes hechos:

  - es invariante frente a transformaciones lineales no singulares de las variables,
  - $d_{Eu}(\mathbf{x}, \mathbf{y}) = d_{Maha}(\mathbf{x}, \mathbf{y})$ cuando la matriz de covarianzas es igual a la matriz identidad; $d_{Pear}(\mathbf{x}, \mathbf{y}) = d_{Maha}(\mathbf{x}, \mathbf{y})$ cuando la matriz de covarianza es igual a $S = diag\left(s_1^2, \ldots, s_l^2\right)$,
  - tiene en cuenta las correlaciones entre las variables.

$$d_{\text{Maha}}(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T S^{-1}(\mathbf{x}-\mathbf{y})} \tag{2.7}$$

Al igual que la anterior, es también invariante frente a cambios de escala.

En la tabla 2.3 se muestra un resumen con las medidas de distancia más utilizadas.

Medida de disimilitud	Ecuación
Distancia de Minkowski	$d_{\text{Mink}}(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{l} \lvert x_i - y_i\rvert^n\right)^{1/n}$
Distancia Euclídea	$d_{\text{Eu}}(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{l} \lvert x_i - y_i\rvert^2\right)^{1/2}$
Distancia de Ciudad o de Manhattan	$d_{\text{Man}}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{l} \lvert x_i - y_i\rvert$
Distancia de Pearson	$d_{\text{Pear}}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{l} \frac{(x_i-y_i)^2}{s_i^2}} = \sqrt{(\mathbf{x}-\mathbf{y})^T S_0^{-1}(\mathbf{x}-\mathbf{y})}$
Distancia de Mahalanobis	$d_{\text{Maha}}(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T S^{-1}(\mathbf{x}-\mathbf{y})}$

Tabla 2.3: Medidas de disimilitud o distancia.

Por lo general, las medidas de distancia se utilizan para aquellas variables o características continuas, mientras que las medidas de similitud son más usadas para variables cualitativas. La selección de una u otra medida depende del problema con el que nos encontremos. Las medidas definidas anteriormente son para conjuntos de datos que se puedan representar como puntos, para datos ordinales, nominales, o incluso mezcla de otros datos. En [25, 26, 11] puede encontrarse más información acerca de estas medidas. Para medidas de similitud en textos, [27] es una buena referencia.

2.2.2. Tipos de agrupamiento.

Dado que existen numerosos problemas en los que el análisis de grupos es necesario, han surgido diversas soluciones que se adecuan a cada una de las necesidades específicas de cada problema. Todos estos algoritmos se pueden clasificar en base a una serie de criterios, dependiendo de cual sea el resultado final, de como se escojan los parámetros de entrada, de la estructura del algoritmo, etc.

A pesar de la gran cantidad de técnicas de agrupamiento existentes en la literatura, todas pueden ser clasificadas en uno de los siguientes cuatro tipos de agrupamiento:

- *Algoritmos de agrupamiento particionales*. Son aquellos que obtienen como resultado una única partición de los datos iniciales, en lugar de una estructura de agrupamiento con varios niveles de particiones.

- *Algoritmos de agrupamiento jerárquicos*. Organizan los datos en estructuras jerárquicas de acuerdo a la matriz de proximidades. Los resultados de estos algoritmos son, por lo general, mostrados en un árbol binario o en un dendograma.

- *Algoritmos de agrupamiento probabilísticos*. Desde el punto de vista probabilístico, se asume que los objetos son generados de acuerdo a algunas distribuciones probabilísticas. Objetos en distintos grupos son generados por distintas distribuciones de probabilidad o son derivados de distintos tipos de funciones de densidad, o de las mismas familias pero con distintos parámetros.

- *Algoritmos de agrupamiento basados en densidades*. Estos algoritmos aplican criterios locales de grupo. Los grupos son tenidos en cuenta como regiones en el espacio de datos de gran densidad de objetos, y los cuales están separados por regiones de menor densidad (ruido). Estas regiones pueden tener cualquier forma y pueden estar distribuidas de cualquier manera.

Por otro lado, debido a que el agrupamiento de datos tiene una aplicabilidad muy extensa y variada, y por tanto resuelve problemas de muy distinta índole, se han de tener en cuenta una serie de temas transversales para la aplicación de estas técnicas. Según [28] la clasificación de todos estos temas transversales se resume en:

- *Jerárquicos o particionales*. Este criterio está relacionado con la estructura del resultado final. Los métodos jerárquicos producen una serie de particiones anidadas mientras que los particionales producen una única partición. La clasificación tradicional de los algoritmos de agrupamiento se basa en este criterio.

- *Aglomerativos o divisivos*. Este aspecto esta relacionado con la estructura algorítmica y la operativa. Un método aglomerativo comienza con cada dato en un grupo distinto, y va sucesivamente mezclando o uniendo grupos hasta que algún criterio de parada se cumpla. Los métodos divisivos, por el contrario, comienzan introduciendo todos los datos en un grupo y van dividiendo sucesivamente este grupo hasta que se cumple con un criterio de parada.

- *Monotéicos o politéicos*. Esta clasificación se refiere al uso secuencial o simultáneo de las características en el proceso de agrupamiento. La mayoría de los algoritmos son politéicos, es decir, que la mayoría de las características se tienen en cuenta a la hora de calcular las distancias entre los datos, y las decisiones son llevadas a cabo en base a estas distancias. En cambio, un algoritmo monotéico considera las características secuencialmente para dividir la colección de datos dada.

- *Duros o difusos (fuzzy)*. Un algoritmo de agrupamiento duro asigna cada objeto a un único grupo durante su operación y a la salida. Por el contrario, un algoritmo de agrupamiento difuso puede asignar a cada objeto o dato a más de un grupo con un cierto grado de pertenencia. Un algoritmo de agrupamiento difuso se puede convertir en uno duro, si asignamos a cada objeto el cluster con mayor grado de pertenencia.

- *Deterministas o estocásticos*. Esta cuestión es más relevante en métodos particionales diseñados para optimizar una función de error cuadrática. Esta optimización puede llevarse a cabo usando técnicas tradicionales o a través de búsquedas aleatorias en el espacio de todas las posibles etiquetas actuales.
- *Incrementales o no incrementales*. Este aspecto se debe tener en cuenta cuando el espacio de datos a ser agrupado es muy grande, y existen limitaciones de memoria o de tiempo de ejecución. Entonces, la arquitectura del algoritmo, puede verse afectada.

2.2.2.1. Agrupamiento particional.

Un algoritmo de agrupamiento particional es aquel que obtiene como resultado una única partición de los datos iniciales, en lugar de una estructura de agrupamiento con varios niveles de particiones. Un algoritmo particional asigna a un conjunto de objetos K grupos sin estructura jerárquica, siendo K un número real menor que el número total de objetos. Este tipo de algoritmos son muy eficientes en aquellas aplicaciones con conjuntos de datos de gran dimensionalidad, pero presentan el problema de que es necesario escoger el número de grupos deseados.

Los algoritmos particionales por lo general producen grupos optimizando una función criterio definida, bien localmente (definida sobre un subconjunto de datos) o bien globalmente (definida sobre el conjunto completo de datos). Una forma de encontrar la partición óptima es a través de una búsqueda combinatoria del conjunto de valores de etiquetas posibles, pero esta solución es computacionalmente inviable. Por lo general, en vez de esta búsqueda prohibitiva, se opta por ejecutar varias veces el algoritmo con distintos puntos de entrada y la mejor configuración obtenida de entre todas las ejecuciones es usada como salida del algoritmo.

Uno de los factores más importantes en los algoritmos particionales es la función criterio. En los algoritmos particionales la función criterio más utilizada es el error cuadrático. La ecuación 2.8 define el criterio de error cuadrático dado un conjunto de entrada $\mathbf{x}_j \in \mathbb{R}^d, j = 1, \ldots, N$ que se quiere agrupar en un conjunto de K grupos, $C = \{C_1, \ldots, C_K\}$; en esta ecuación, $\mathbf{\Gamma}$ es una matriz de particiones y $\mathbf{M}$ es una matriz de medias, centroides o prototipos de grupos. Los grupos resultantes de esta función de error cuadrática son frecuentemente denominados *particiones de varianza mínima* según se explica en [14].

$$J\left(\mathbf{\Gamma}, \mathbf{M}\right) = \sum_{i=1}^{K} \sum_{j=1}^{N} \gamma_{ij} \left\|\mathbf{x}_j - \mathbf{m}_i\right\|^2 \tag{2.8}$$

Donde $\mathbf{\Gamma} = [\gamma_{ij}]$ es la matriz de elementos, siendo

$$\gamma_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_j \in \text{cluster } i \\ 0 & \text{otro} \end{cases}$$

Con

$$\sum_{i=1}^{K} \gamma_{ij} = 1 \; \forall \; j;$$

y donde $\mathbf{M} = [\mathbf{m}_1, \ldots, \mathbf{m}_K]$ es la matriz de medias, siendo $\mathbf{m}_i = \frac{1}{N_i} \sum_{j=1}^{N} \gamma_{ij} \mathbf{x}_j$ una muestra de la media del grupo i dados N_i objetos en ese grupo.

Existen numerosos algoritmos particionales basándose en distintos aspectos, como por ejemplo: en las funciones criterio, en los atributos de entrada, en el tipo de salida, etc. Dos de los algoritmos más utilizados y más sencillos son: *K-Means Clustering* [14, 29] y *Fuzzy C-Means Clustering*. Ambos se basan en el mismo principio fundamental. Pero, el primero es un algoritmo duro, es decir, a cada objeto se le asigna un único grupo, mientras que el segundo es difuso o *fuzzy*, es decir, un mismo objeto es asignado a distintos grupos. Ambos algoritmos son explicados con más detenimiento a continuación.

Agrupamiento K-Means. Este algoritmo es el más conocido de entre los algoritmos de agrupamiento particionales y usa como función criterio una función de error cuadrático. El algoritmo 1 muestra el pseudocódigo de este método.

Algorithm 1 Algoritmo de agrupamiento *K-Means*

1: Inicializar aleatoriamente las K-particiones. Calcular $\mathbf{M} = [\mathbf{m}_1, \ldots, \mathbf{m}_K]$;
2: **while** $\exists$ cambios en $\mathbf{M}$ **do**
3: clasificar ojectos de acuerdo al $\mathbf{m}_i$ más cercano;
4: recalcular $\mathbf{M}$ basado en el actual;
5: **end while**

Salida: $\mathbf{M}$.

Existen diversas maneras de asignar cada objeto del conjunto de datos al grupo C_w más cercano, es decir, de realizar el paso 3 del algoritmo. Una de las técnicas más usadas es que un objeto $\mathbf{x}_j$ pertenezca al grupo C_w si $\|\mathbf{x}_j - \mathbf{m}_w\| < \|\mathbf{x}_j - \mathbf{m}_i\|$ para $j = 1, \ldots, N$, $i \neq w$ e $i, w = 1, \ldots, K$.

El algoritmo K-Means es muy simple y es fácil de implementar para resolver muchos problemas prácticos. Funciona bien para grupos compactos y de forma hiperesférica. La complejidad de este algoritmo es $O\left(NKd\right)$, y teniendo en cuenta que K y d son por lo general menores que N, K-Means se puede usar en conjuntos de datos de gran dimensionalidad o de muchos datos. Una forma de acelerar el algoritmo es utilizando técnicas de paralelismo a nivel de software [30]. Pero K-Means también tiene algunas desventajas y un gran número de variantes de este algoritmo han surgido para resolver estas desventajas. Las principales desventajas y algunas de sus soluciones son:

- No existe un método eficiente y universal para identificar las particiones iniciales y para determinar el número de grupos K. La convergencia de los centroides varía con distintos puntos de entrada. Una estrategia para este problema es ejecutar el algoritmo varias

veces para particiones iniciales aleatorias. El estudio llevado a cabo por Peña, Lozano y Larrañaga en [31] compara este método aleatorio con varias técnicas de inicialización.

- El procedimiento óptimo iterativo de K-Means no puede garantizar la convergencia a un óptimo global. Las técnicas óptimas estocásticas, como por ejemplo los algoritmos genéticos, pueden encontrar este óptimo global, pero con un gran coste computacional.

- El algoritmo K-Means es sensible al ruido y a valores atípicos (*outliers*). Para solucionar este problema surgieron los algoritmos ISODATA [32], PAM [25] y *K-Medoids* [33].

- La definición de «media» (*mean*) limita la aplicación de este algoritmo únicamente a problemas con variables numéricas. El algoritmo *K-medoids*, previamente nombrado, soluciona esta limitación.

Además de los algoritmos nombrados que solucionan algunos de los problemas y limitaciones que presenta el algoritmo de K-Means, existen más algoritmos que utilizan la base del K-Means pero con mejoras notables.

Agrupamiento Fuzzy C-Means. En todas las iteraciones del algoritmo *K-Means*, cada objeto es asignado a un único cluster. Otra posibilidad que existe es la de relajar esta condición y asumir que cada objeto $\mathbf{x}_j$ tiene algún grado de pertenencia, $\mu_i(\mathbf{x}_i)$, al cluster C_i, con $0 \leq \mu_i(\mathbf{x}_i) \leq 1$. El algoritmo resultante de esta relajación es el denominado *Fuzzy C-Means* y busca minimzar la función de coste global:

$$J(\mathbf{U}, \mathbf{M}) = \sum_{i=1}^{c} \sum_{j=1}^{N} \mu_{ij}^{b} \|\mathbf{x}_j - \mathbf{m}_i\|^2$$

siendo $\mathbf{U} = [\mu_{ij}]_{c \times N}$ la matriz de particiones borrosa, $\mu_{ij} \in [0, 1]$ el grado de pertenencia del objeto $\mathbf{x}_j$ al grupo C_i, $\mathbf{M} = [\mathbf{m}_i, \ldots, \mathbf{m}_c]$ la matriz de prototipos de grupos y $b > 1$ un parámetro de elección libre escogido para ajustar la «mezcla» de los distintos grupos.

Las probabilidades de pertenencia de los objetos a los grupos son normalizadas según la ecuación: $\sum_{i=1}^{c} \hat{P}(C_i|\mathbf{x}_j) = 1, j = 1, \ldots, N$. El mínimo de la función de coste se obtiene cuando $\partial J / \partial \mathbf{m}_i = 0$ y cuando $\partial J / \partial \hat{P}_j = 0$ llevándonos a las condiciones:

$$\mathbf{m}_j = \frac{\sum_{i=1}^{N} [P(C_i|\mathbf{x}_j)]^b \mathbf{x}_j}{\sum_{i=1}^{N} [P(C_i|\mathbf{x}_j)]^b} \tag{2.9}$$

y la probabilidad de pertenencia de un objeto,$\mathbf{x}_j$, a un grupo, C_i:

$$P\left(C_i|\mathbf{x}_j\right) = \frac{\left(1/\left\|\mathbf{x}_j - \mathbf{m}_i\right\|^2\right)^{1/(b-1)}}{\sum_{r=1}^{c}\left(1/\left\|\mathbf{x}_j - \mathbf{m}_r\right\|^2\right)^{1/(b-1)}} \tag{2.10}$$

En general, la función de coste es minimizada, cuando los centros de los grupos, $\mathbf{m}_j$, están cerca de aquellos puntos que tienen una elevada probabilidad de pertenecer al grupo C_j. Las medias de los grupos y las probabilidades de los puntos se estiman iterativamente según el algoritmo 2.

Algorithm 2 Algoritmo de agrupamiento *Fuzzy C-Means*

1: Inicializar las K-particiones aleatoriamente. Calcular $P\left(C_i|\mathbf{x}_j\right)$ y $\mathbf{M} = [\mathbf{m}_1, \ldots, \mathbf{m}_c]$;
2: normalizar $P\left(C_i|\mathbf{x}_j\right)$;
3: **while** $\exists$ cambios en $\mathbf{M}$ y en $P\left(C_i|\mathbf{x}_j\right)$ **do**
4: clasificar objetos según el $\mathbf{m}_i$ más cercano;
5: recalcular $\mathbf{M}$ según la ecuación 2.9;
6: recalcular $P\left(C_i|\mathbf{x}_j\right)$ según la ecuación 2.10;
7: **end while**

Salida: $\mathbf{M}$.

Aparentemente este método mejora el problema de la convergencia del algoritmo *K-Means*, pero posee algunas desventajas. La principal desventaja que presenta este método es que para normalizar las probabilidades de que un objeto $\mathbf{x}_j$ pertenezca a un grupo C_i depende implícitamente del número de grupos, y si este número se especifica de manera errónea aparecerán graves problemas. Y sin embargo, al igual, que con el algoritmo *K-Means*, han surgido otros algoritmos que tratan de solucionar estas desventajas.

2.2.2.2. Agrupamiento jerárquico.

Los algoritmos de agrupamiento jerárquicos organizan los datos en estructuras jerárquicas de acuerdo a la matriz de proximidades. El proceso de agrupamiento comienza suponiendo que existe una secuencia de particiones de n objetos en K grupos. La primera de estas particiones es una partición en n grupos, cada uno conteniendo exactamente un objeto. La siguiente es una partición en $n-1$ grupos, la siguiente en $n-2$, y así sucesivamente hasta la partición n-ésima en la cual, todos los objetos forman un grupo. Se dice que nos encontramos en el nivel c de una secuencia cuando $K = n - c + 1$. Así, el primer nivel corresponde a n grupos y el nivel n corresponde a un grupo. Dados dos objetos x_1 y x_2, en algún nivel estarán agrupados juntos en el mismo grupo. Si esta secuencia cumple la propiedad de que siempre que dos objetos se encuentran en un mismo grupo en el nivel c y que siguen estando juntos en el resto de niveles, entonces esta secuencia es un agrupamiento jerárquico.

Los resultados de estos algoritmos son, por lo general, mostrados en un árbol binario o en un dendograma. El nodo raíz del dendograma representa el conjunto entero de datos y cada nodo hoja se considera como un objeto. Los nodos intermedios describen de qué modo los objetos están próximos entre sí. La altura del dendograma normalmente representa la distancia entre cada par de objetos o grupos a un objeto y un grupo. Los resultados del algoritmo de agrupamiento se obtienen de realizar cortes en distintos niveles del dendograma. La figura 2.7 representa un ejemplo de dendograma de un estudio realizado sobre las mujeres gestantes en América [34]. La gran ventaja de estos dendogramas es que ofrecen descripciones muy informativas y visuales acerca de las posibles estructuras de los grupos de datos.

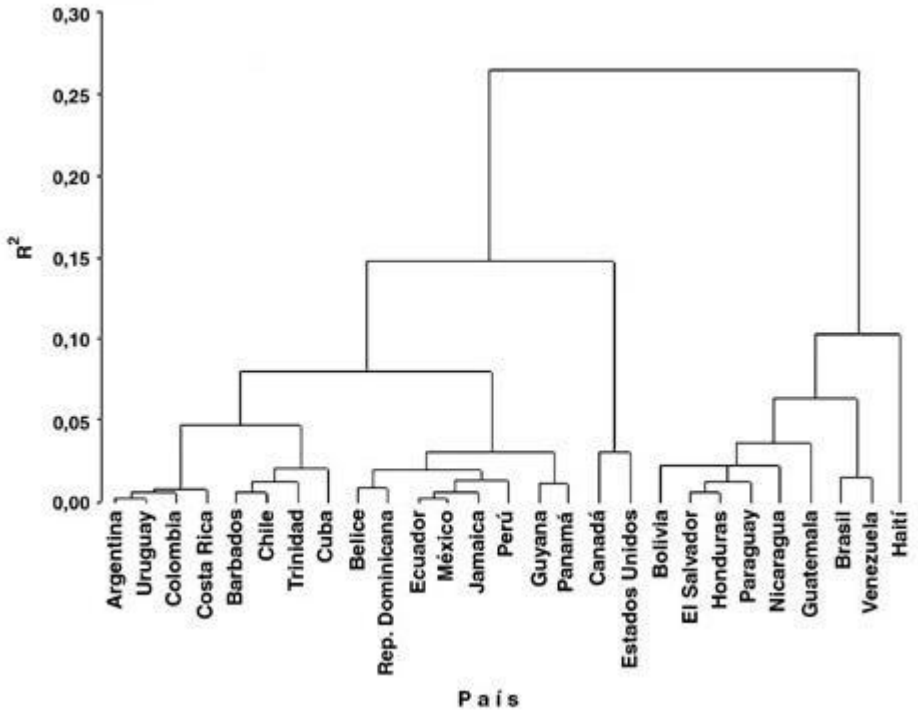


Figura 2.7: Dendograma sobre un estudio de mujeres gestantes en América.

Atendiendo a la estructura algorítmica, los algoritmos jerárquicos se pueden dividir en dos: aglomerativos y divisivos. Los algoritmos aglomerativos comienzan por n grupos y en cada uno de ellos se incluye exactamente un objeto. A medida que se va ejecutando el algoritmo, los grupos se van mezclando hasta que se llega a un único grupo que incluye a todos los objetos. Los algoritmos divisivos proceden de manera contraria. En un principio, el conjunto entero de datos pertenece a un único grupo y sucesivamente se va dividiendo hasta que todos los grupos contienen un único objeto, es decir, hasta que se forman tantos grupos como objetos. El problema que surge de los algoritmos divisivos, es que para un conjunto de n objetos, existen $2^{n-1} - 1$ posibles divisiones, lo cual supone un coste computacional muy elevado. Por esta razón, los algoritmos divisivos no son muy utilizados.

La mayoría de los pasos a seguir en un algoritmo de agrupamiento jerárquico aglomerativo se pueden contemplar en el algoritmo 3, donde c es el número esperado de grupos finales.

Algorithm 3 Agrupamiento jerárquico aglomerativo

1: Inicializar $c, \hat{c} \leftarrow n$, $D_i \leftarrow x_i, i = 1, \ldots, n$;
2: **repeat**
3: $\quad \hat{c} \leftarrow \hat{c} - 1$;
4: $\quad$ Encontrar el grupo más cercano, digamos D_i y D_j;
5: $\quad$ Mezclar D_i y D_j;
6: **until** $c = \hat{c}$

Salida: c grupos o *clusters*.

El algoritmo descrito termina cuando el número especificado de grupos se haya obtenido, y entonces se devuelven estos grupos, descritos como conjuntos de puntos en vez de como una media o un vector. Si continuamos hasta que $c = 1$ entonces podemos construir un dendograma como el de la figura 2.7. En el paso cuatro del algoritmo, se utilizan métricas o medidas de distancia. En cualquier nivel, la distancia entre los grupos más cercanos puede dar un valor de la disimilitud en ese nivel. Dependiendo de la medida de la distancia utilizada se tienen distintos algoritmos.

Cuando la medida de distancia entre grupos es $d_{min}\left(D_i, D_j\right) = \min_{\mathbf{x}\in D_i, \mathbf{x}'\in D_j} \|\mathbf{x} - \mathbf{x}'\|$ el algoritmo se denomina *nearest-neighbour cluster algorithm* o algoritmo mínimo. Si el algoritmo termina cuando la distancia entre los grupos más cercanos excede un determinado umbral, entonces se denomina *single-linkage algorithm*. Este algoritmo es capaz de generar un árbol de expansión de un grafo conexo y no dirigido (*spanning tree*).

Cuando la medida de distancia usada es $d_{max}\left(D_i, D_j\right) = \max_{\mathbf{x}\in D_i, \mathbf{x}'\in D_j} \|\mathbf{x} - \mathbf{x}'\|$ el algoritmo se denomina *farthest-neighbour cluster algorithm* o algoritmo máximo. Si el algoritmo termina cuando la distancia entre los grupos más cercanos excede un determinado umbral, entonces se denomina *complete-linkage algorithm*. Una aplicación de este algoritmo es la de generar subgrafos completos.

El principal problema común a todos estos algoritmos es que son muy sensibles al ruido o a pequeños cambios en las posiciones de los puntos. Una vez que un objeto es asignado a un grupo, no volverá a ser tenido en cuenta, con lo cual el algoritmo no es capaz de corregir posibles errores en la clasificación.

La complejidad de la mayoría de estos algoritmos es de $O(N^2)$ y su elevado coste limita la aplicación a conjuntos de datos de gran dimensionalidad. Otra desventaja es la tendencia a formar grupos esféricos.

Últimamente, con la necesidad de usar conjuntos de datos de tamaños más grandes han surgido nuevas técnicas jerárquicas que permiten su utilización, como por ejemplo: CURE [35], BIRCH [36] o ROCK [37];

2.2.2.3. Agrupamiento probabilístico o *Mixture Densities-Based clustering*.

Desde el punto de vista probabilístico, se asume que los objetos son generados de acuerdo a algunas distribuciones de probabilidad. Objetos en distintos grupos son generados siguiendo distintas distribuciones de probabilidad, estos pueden haber sido derivados de distintos tipos de funciones de densidad o de las mismas familias pero con distintos parámetros. Si las distribuciones son conocidas, encontrar grupos de un conjunto de datos será equivalente a estimar los parámetros de algunos modelos. Supongamos que conocemos la probabilidad a priori,$P\left(C_i\right)$, de un grupo $C_i, i = 1, \ldots, K$, siendo K conocida, y la densidad de probabilidad condicional $p\left(\mathbf{x}|C_i, \theta_i\right)$, donde θ_i es el vector de parámetros desconocido. Entonces, la densidad de probabilidad mezclada para todo el conjunto de datos se puede expresar como:

$$p\left(\mathbf{x}|\theta_i\right) = \sum_{i=1}^{K} p\left(\mathbf{x}|C_i, \theta_i\right) P\left(C_i\right)$$

donde $\theta = (\theta_1, \ldots, \theta_K)$ y $\sum_{i=1}^{K} P\left(C_i\right) = 1$. Mientras que el vector θ es decidido, la probabilidad a posteriori de asignar cada objeto a un grupo se puede calcular fácilmente a través del teorema de Bayes.

Para estimar los parámetros, el estimador de máxima verosimilitud es considerado un método estadístico muy importante y es considerado el mejor estimador que maximiza la probabilidad de generar todas las observaciones.

Desafortunadamente, dado que en la mayoría de los casos, las soluciones de las ecuaciones de verosimilitud no pueden obtenerse de manera analítica, métodos subóptimos iterativos son usados para estimadores de máxima verosimilitud. De entre estos métodos, el algoritmo EM (*expectation-maximization*) es el más conocido. Éste considera el conjunto de datos como incompleto y divide cada punto $\mathbf{x}_j$ en dos partes, $\mathbf{x}_j = \left\{\mathbf{x}_j^g, \mathbf{x}_j^m\right\}$, donde $\mathbf{x}_j^g$ representa las características observables y $\mathbf{x}_j^m = \left(x_{j1}^m, \ldots, x_{jK}^m\right)$ son las características desconocidas, teniendo en cuenta que x_{ji}^m toma valores 1 o 0 según $\mathbf{x}_j^m$ pertenezca o no a la componente i. El algorimo EM estandar según los pasos mostrados en 4 genera una serie de parámetros estimados $\left\{\theta^0, \theta^1, \ldots, \theta^T\right\}$, donde T representa en qué instante se alcanza la convergencia del criterio. Puede encontrarse información más detallada acerca de este algoritmo en [38].

Algorithm 4 Algoritmo EM estandar

1: Inicializar θ^0 y $t = 0$;
2: **repeat**
3: paso E: Calcular $Q\left(\theta^0, \theta^t\right) = E\left[\log p\left(\mathbf{x}^g, \mathbf{x}^m|\theta|\mathbf{x}^g, \theta^t\right)\right]$;
4: paso M: $\theta = argmax_\theta Q\left(\theta, \theta^t\right)$;
5: t = t+1;
6: **until** Se satisfaga la condición de convergencia

Salida: θ.

Las principales desventajas del algoritmo EM son: la sensibilidad a la elección de los parámetros de entrada, el efecto de una matriz de covarianzas singulares, la posibilidad de converger

a un óptimo local y la baja tasa de convergencia. Variantes de este método han surgido para solucionar algunos de estos problemas.

2.2.2.4. Agrupamiento basado en densidades.

Los algoritmos basados en densidades intentan encontrar grupos de datos teniendo en cuenta la distribución de los puntos, de tal forma que los grupos que se forman tienen una alta concentración de puntos en su interior mientras que entre ellos aparecen zonas de baja densidad.

Estos algoritmos usan diversas técnicas para determinar dichos grupos, entre las que se pueden encontrar: técnicas basadas en grafos, basadas en histogramas, en kernels, aplicando la regla K-NN, empleando los conceptos de punto central, borde o ruido, etc. Entre ellos podemos mencionar los algoritmos DBSCAN [2], KNNCLUST [39], SNN [40], DBCLASD [41] o DENCLUE [42].

El primer algoritmo que emplea este enfoque para dividir el conjunto de datos es DBSCAN, y en él aparecen los conceptos de punto central, borde y ruido, los cuales son empleados para determinar los diferentes grupos. Otros algoritmos basados en densidad que siguen la línea de DBSCAN son: OPTICS [43] y GDBSCAN [44].

Dada la creciente importancia de estos métodos y el gran número de aplicaciones de agrupamiento por describir, este trabajo se centra en el estudio de estas técnicas de agrupamiento de densidades. Este estudio se realizará en el capítulo 3 con una descripción de las técnicas y en el 4 con la realización de un estudio comparativo.

2.2.3. Aplicaciones.

Los algoritmos de agrupamiento pueden aplicarse en muchos campos. Como por ejemplo, en marketing, buscando grupos de clientes con comportamientos similares entre una gran base de datos con información relativa a los clientes; en Biología, clasificando plantas y animales a partir de sus características; en Bibliotecas, ordenando libros; en compañías aseguradoras, identificando grupos de pólizas de seguros con un gran coste, o fraudes; en planificación de las ciudades, identificando grupos de casas según su tipo, su localización o su valor; en estudios de terremotos, identificando zonas de alto riesgo a partir de datos observados de otros terremotos; en Internet, clasificando documentos.

Otras aplicaciones muy interesantes de las técnicas de agrupamiento son, por ejemplo, la segmentación de imágenes [45, 3] y el reconocimiento de caracteres u objetos.

La mayoría de las aplicaciones previamente nombradas sirven para ayudar a comprender los datos o para mejorar su entendimiento. Pero también existe la posibilidad de utilizar estos grupos de manera práctica. En [26] se explica como se pueden resumir en cuatro direcciones

básicas el uso de técnicas de agrupamiento para buscar alguna utilidad en los datos. Estas direcciones básicas son:

- *Reducción de datos*. En numerosas ocasiones, la cantidad de datos disponible es enorme y hace que su manejo y procesado sea muy complicado. El análisis de grupos puede ser utilizado para agrupar los datos en un número sensible de grupos de manera que cada grupo se procese como una entidad. Este tipo de aplicación se da, por ejemplo, en la transmisión de datos donde los datos se comprimen antes de ser enviados.
- *Generación de hipótesis*. En este caso, se aplica el análisis de grupos al conjunto de datos para deducir alguna hipótesis sobre lo que concierne a la naturaleza de los datos. Estas hipótesis han de ser verificadas usando otros conjuntos de datos.
- *Test de hipótesis*. Otra posibilidad es utilizar el análisis de grupos para validar hipótesis hechas sobre el conjunto de datos.
- *Predicción basada en grupos*. Podemos aplicar el análisis de grupos al conjunto de datos, y los grupos resultantes son caracterizados según aquellas propiedades de aquellos datos de los que están formados. En consecuencia, si tenemos un patrón desconocido podemos predecir a qué grupo pertenece y caracterizarlo de acuerdo a la caracterización del grupo respectivo. Un ejemplo de esta aplicación es en el campo de la medicina o la psicología.

Capítulo 3

Algoritmos de agrupamiento basados en densidades.

Tal como se ha explicado anteriormente, existen un gran número de algoritmos de agrupamiento, cada uno adaptado a las necesidades específicas de los distintos problemas. En este apartado hablaremos de uno de los tipos de algoritmos de agrupamiento surgido recientemente y de gran utilidad. Son los denominados algoritmos de agrupamiento basados en densidades.

En la primera parte de este capítulo se describen los algoritmos de agrupamiento basados en densidades, se estudian sus particularidades y los motivos de su utilización frente a otros algoritmos clásicos como K-Means.

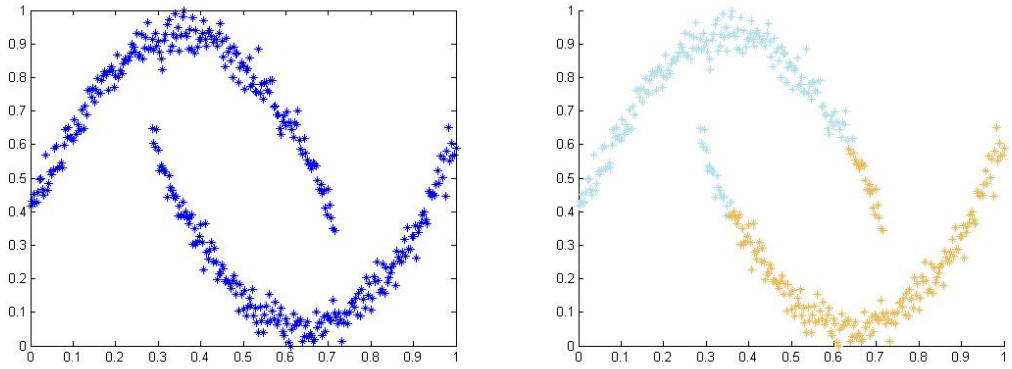
A continuación, se describen en detalle algunas de las técnicas más conocidas dentro de este tipo de algoritmos. Uno de los primeros algoritmos desarrollados dentro de este tipo es el denominado DBSCAN, basado en la unión de zonas de alta densidad. Posteriormente, se desarrolló una nueva versión de este algoritmo (DBSCAN-4C [46]) donde se añade información sobre las correlaciones entre datos.

A continuación, se estudia el algoritmo de agrupamiento basado en el desplazamiento de medias (*Mean Shift* [47]). Este algoritmo es comúnmente aplicado en técnicas de segmentación de imágenes [47] y recientemente para la selección de características [1]. Por último, se define un nuevo algoritmo de agrupamiento basado en la búsqueda de curvas locales principales y que usa el mismo principio que el algoritmo *mean shit*.

En este capítulo, y con el propósito de mejorar el entendimiento y la comprensión de los distintos algoritmos, e igualmente facilitar la visualización de los resultados, se definen cinco conjuntos de datos. Dos de ellos consisten en dos grupos con forma de curva cuadrática; el primer conjunto de datos no contiene ruido, mientras que el segundo sí. El tercer conjunto de datos está conformado por dos grupos que siguen una distribución gaussiana. El cuarto conjunto de datos está definido en tres dimensiones y contiene tres grupos: dos lineales y una esfera normalmente distribuida. El quinto y último de los conjuntos de datos utilizados en este capítulo contiene dos grupos donde el primero tiene forma de espiral y el segundo cuadrática.

3.1. Definición de los algoritmos de agrupamiento basados en densidades.

Por lo general, si una persona tiene la posibilidad de observar un conjunto de datos, le resulta fácil identificar a primera vista grupos de puntos y/o puntos que son ruido y/o que no pertenecen a ningún grupo. La principal razón por la que reconocemos estos grupos es porque dentro de cada uno existe una densidad de puntos que es considerablemente mayor a la existente fuera de ese grupo. Además, la densidad de puntos entre áreas de ruido es menor que la densidad en cualquiera de los otros grupos. Este fenómeno puede observarse en la figura 3.1a. Muchos de los algoritmos de agrupamiento tradicionales no son capaces de integrar este resultado. Si introducimos el conjunto de datos mostrado en la figura 3.1a, donde se puede apreciar claramente que existen dos grupos, al algoritmo de agrupamiento *K-Means*, se obtiene el resultado mostrado en la figura 3.1b, donde cada grupo está representado por un color distinto. Como puede observarse el resultado obtenido dista mucho del esperado.



(a) Conjunto de datos 1

(b) Resultado de aplicar K-Means

Figura 3.1: Conjunto de datos 1 y resultado de la ejecución del algoritmo K-Means.

Para solucionar este tipo de problemas y teniendo en cuenta el fenómeno explicado anteriormente, surgieron los algoritmos basados en densidades. Estos enfocan el problema de la búsqueda de grupos en un conjunto de datos teniendo en cuenta la distribución de densidad de los puntos, de tal forma que los grupos que se forman tienen una alta concentración de puntos en su interior mientras que entre ellos aparecen zonas de baja densidad.

Existen numerosos algoritmos basados en densidades donde cada uno utiliza una técnica distinta. Por ejemplo basándose en grafos, en histogramas, en la regla KNN, etc. y que surgen de las necesidades específicas de cada problema.

Dentro de los algoritmos basados en densidades se puede destacar un grupo de métodos, que son denominados métodos basados en mallas, *grid-based methods*. Una de las características a resaltar de estos métodos es su rapidez, debido a que su tiempo de procesamiento depende del tamaño de las rejas y no tanto del número de objetos. Estos métodos usan una reja o malla

para hacer una partición en celdas del espacio. Los objetos que se encuentran en cada celda son representados por un conjunto de atributos estadísticos de dicha celda. El agrupamiento se lleva a cabo utilizando la información estadística de cada celda, en vez de usar todo el conjunto de datos. Dado que el tamaño de la reja es mucho menor que el número de objetos, la velocidad de procesamiento se ve incrementada. Sin embargo, para distribuciones de datos muy concentrados o irregulares, se necesita una reja con una granularidad muy alta para obtener un resultado óptimo, es decir, para descubrir grupos muy próximos, por lo que su ventaja en rapidez se ve atenuada. Ejemplos de estos algoritmos son: STING [48], WaveCluster [49] y CLIQUE [50], que usan rejas uniformes, o MAFIA [51] y el método de agrupamiento basado en AMR [52] que usan rejas adaptables.

Una vez explicada la motivación de los algoritmos basados en densidades, podemos centrarnos en estudiar alguno de estos algoritmos.

3.2. DBSCAN.

Los algoritmos basados en agrupamientos por densidades localizan regiones de alta concentración de puntos que se encuentran separadas entre sí por regiones con una menor densidad. DBSCAN es un algoritmo simple basado en densidades, que trata de ilustrar una serie de conceptos necesarios para cualquier algoritmo basado en densidades.

El algoritmo DBSCAN [2] fue desarrollado en el año 1996 por M. Ester et al., en la Universidad de Munich. Este algoritmo se basa en la búsqueda de puntos centrales o *core* en los grupos. Estos puntos centrales poseen un área o región de vecindad para un radio determinado que contiene al menos un número mínimo de puntos, es decir, que su área de vecindad excede un umbral determinado.

Este método es muy sencillo de implementar, pero la densidad de los puntos depende del radio de la región de vecindad especificado. De este modo, si el radio es suficientemente grande todos los puntos tendrán una densidad igual al número de puntos total del conjunto de datos. Por el contrario, si es muy pequeño todos los puntos tendrán una densidad igual a 1, es decir, el punto se encontrará aislado.

Algunas de las aplicaciones de DBSCAN realizadas con éxito son: la detección de usos en las tierras a partir de imágenes satélite, la creación de perfiles de usuarios en Internet mediante la agrupación de sesiones Web, o el agrupamiento de bases de datos de imágenes en histogramas en color facilitando la búsqueda de imágenes similares [53].

En el siguiente apartado, se explican los fundamentos básicos para la implementación de DBSCAN, cuyo algoritmo se desarrolla en el apartado 3.2.2.

3.2.1. Fundamentos.

La idea principal del algoritmo DBSCAN es encontrar todos los puntos centrales, donde los puntos centrales de un grupo son aquellos que tienen una región de vecindad que contiene un número mínimo de puntos para un radio determinado. La forma de la región de vecindad viene determinada por la elección de la medida de la distancia entre dos puntos, $dist\,(p,q)$. Por ejemplo si se utiliza la distancia de Manhattan, DBSCAN tiende a formar grupos rectangulares.

La *región de vecindad* de un punto p perteneciente a una base de datos D dado un radio Eps se define como:

$$N_{Eps}\,(p) = \{q \in D | dist\,(p,q)\} \leq Eps \tag{3.1}$$

Teniendo en cuenta esta definición podríamos dar una primera versión del algoritmo buscando si para cada punto se cumple con el criterio de mínimo número de puntos en la vecindad; pero este algoritmo no funcionaría, ya que en general en los bordes de los grupos las densidades no son las mismas que en el interior de estos.

De este modo en un conjunto de datos se pueden distinguir tres tipos de puntos:

- *Puntos centrales o core*. Son aquellos puntos que se encuentran en el interior de un grupo. Un punto es central si el número de puntos en la región de vecindad definida según el radio Eps excede un cierto umbral conocido, $MinPts$. Es decir, si se cumple $|N_{Eps}\,(p)\,| \geq MinPts$.
- *Puntos frontera o de borde*. Estos puntos no son puntos centrales, pero pertenecen a la región de vecindad de uno o más puntos centrales.
- *Puntos de ruido*. Estos puntos no son ni centrales ni frontera, es decir, no pertenecen a la región de vecindad de ningún otro punto.

Para que el algoritmo funcione correctamente, es necesario que para cada punto p, de un grupo C, exista otro punto q en C, tal que p esté dentro de la región de vecindad de q, y que esta región de vecindad contenga un número mínimo de puntos. De este modo, se define:

Definición 1 Un punto p es *densamente alcanzable de manera directa (directly density-reachable)* desde un punto q dados el radio Eps y el número mínimo de puntos $MinPts$, si:

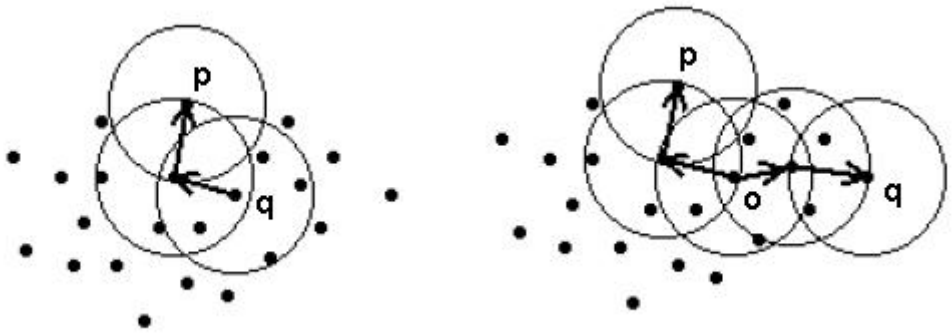
1. $p \in N_{Eps}\,(q)$, esto es, p pertenece a la región de vecindad de q, y
2. $|N_{Eps}\,(q)| \geq MinPts$, es decir q es un punto central [2].

Definición 2 Un punto p es *densamente alcanzable (density-reachable)* desde un punto q dados el radio Eps y el número mínimo de puntos $MinPts$, si existe una cadena de puntos $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ tal que p_{i+1} es densamente alcanzable de manera directa desde p_i [2].

Densamente alcanzable es una extensión canónica de densamente alcanzable de manera directa. Esta relación es transitiva, pero no es simétrica. La relación de densamente alcanzable es simétrica para puntos centrales. Dos puntos frontera de un mismo grupo, pueden no ser densamente alcanzables entre ellos, ya que es posible que la condición de que sean centrales no se cumpla para ambos. No obstante, debe existir un punto central en el grupo C desde el cual los puntos frontera sean densamente alcanzables. Con esto se puede introducir la siguiente definición:

Definición 3 Un punto p está *densamente conectado (density-connected)* a un punto q dados el radio Eps y el número mínimo de puntos $MinPts$, si existe un punto o tal que ambos, p y q, son densamente alcanzables desde o dados Eps y $MinPts$. La relación densamente conectado es una relación simétrica y reflexiva [2].

Para comprender mejor estas definiciones se pueden observar las siguientes figuras. En la figura 3.2a, se muestra como un punto p es densamente alcanzable desde otro punto q, mientras que q no es densamente alcanzable desde p, puesto que la región de vecindad de p no contiene el número mínimo de puntos. La figura 3.2b obtenida de [2], muestra como p y q están densamente conectados desde el punto o.



(a) Punto densamente alcanzable. (b) Puntos densamente conectados.

Figura 3.2: Puntos densamente alcanzables y densamente conectados.

Una vez establecidos los conceptos de alcanzabilidad y de conectividad podemos pasar a definir formalmente los conceptos de grupo y ruido dados el conjunto de datos D, el radio de la región de vecindad Eps y el número mínimo de puntos $MinPts$.

Definición 4 Sea un conjunto de datos D, se dice que un *grupo* C dados el radio Eps y el número mínimo de puntos $MinPts$, es un subconjunto no vacío de D que satisface las siguientes condiciones:

1. $\forall p, q$: si $p \in C$ y q es densamente alcanzable desde p, dados Eps y $MinPts$, entonces $q \in C$. (Maximitud)

2. $\forall p, q \in C$: q está densamente conectado a p dados Eps y $MinPts$. (Conectividad) [2].

Definición 5 Sean $C_1, \ldots, C_K$ los grupos del conjunto de datos D dados los parámetros Eps_i y $MinPts_i$, $i = 1, \ldots, K$. Entonces, se define el *ruido* como aquellos datos pertenecientes a D que no forman parte de ningún grupo C_i, es decir: $ruido = \{p \in D | \forall i : p \notin C_i\}$ [2].

Cabe destacar que un grupo C, dados Eps y $MinPts$, contiene al menos un número mínimo de puntos, $MinPts$ por las siguientes razones. Dado que C contiene al menos un punto p, p debe estar densamente conectado a sí mismo vía otro punto o (el cual puede ser el mismo punto p). Por lo tanto, al menos el punto o debe satisfacer la condición de punto central y, en consecuencia, la región de vecindad de o contiene al menos $MinPts$ puntos.

Los siguientes lemas son necesarios para validar la corrección del algoritmo de agrupamiento que se quiere definir. En general, estos lemas intentan contemplar que, dados los parámetros Eps y $MinPts$, se puede descubrir un grupo si se siguen dos pasos fundamentales: primero, se escoge como comienzo un punto arbitrario del conjunto de datos que satisfaga la condición de ser un punto central y a continuación, se recuperan todos los puntos que son densamente alcanzables desde este punto inicial obteniendo así el grupo que lo contiene.

Lema 3.2.1 Sea p un punto en D y $|N_{Eps}(p)| \geq MinPts$. Entonces el conjunto $O = \{o | o \in D \; y \; o$ es densamente alcanzable desde p dados $Eps \; y \; MinPts\}$ es un grupo dados Eps y $MinPts$ [2].

No es algo obvio el que un grupo, C, conocidos Eps y $MinPts$, este únicamente determinado por cualquiera de sus puntos centrales. Sin embargo, cada punto en C es densamente alcanzable desde cualquiera de los puntos centrales de C y, entonces, un grupo C contiene exactamente los puntos que son densamente alcanzables desde cualquier punto central de C.

Lema 3.2.2 Sea C un grupo tal que son conocidos Eps y $MinPts$, y sea p cualquier punto en C tal que $|N_{Eps}(p)| \geq MinPts$. Entonces C es igual al conjunto definido por: $O = \{o | o \in D \; y \; o$ es densamente alcanzable desde p, dados $Eps \; y \; MinPts\}$ [2].

3.2.2. Algoritmo.

En el apartado anterior se han planteado las bases necesarias para formular el algoritmo DBSCAN. Este algoritmo descubre grupos y ruido en un conjunto de datos según las definiciones dadas anteriormente de estos conceptos. Como parámetros de entrada al algoritmo es necesario definir el valor del radio de la región o del área de vecindad, Eps, y el número mínimo de puntos, $MinPts$.

Para encontrar los grupos, DBSCAN comienza con un punto p arbitrario y se van recopilando todos los puntos que son densamente alcanzables desde p con respecto a Eps y $MinPts$. Si p es un punto central, entonces el proceso terminará formando un grupo con respecto a Eps y $MinPts$ según se vio en el segundo lema. Si por el contrario, p es un punto frontera o es ruido no hay puntos que sean densamente alcanzables desde p y, en este caso, p se clasifica como ruido y se pasa al siguiente punto del conjunto total. Si un punto p es etiquetado inicialmente como ruido, pero después se descubre que es densamente alcanzable desde otro punto o, entonces se le quita la etiqueta de ruido y se le asigna la de un punto frontera, clasificándolo en el mismo grupo que el punto o desde el que es densamente alcanzable. El proceso completo para la agrupación usando DBSCAN se muestra en el algoritmo 5.

Al finalizar el algoritmo, se obtienen dos vectores del mismo tamaño que el conjunto de datos: el vector *class* y el vector *type*. El vector *class* contiene la asignación de cada una de las instancias del conjunto de datos al grupo correspondiente, mientras que el vector *type* define el tipo de punto que es cada instancia, es decir, si es central, frontera o ruido.

La complejidad de este algoritmo es $O\left(n \times \text{tiempo para encontrar puntos en la vecindad}\right)$, siendo n el número de puntos del conjunto de datos. En el peor de los casos, la complejidad será de $O\left(n^2\right)$. Si los datos están estructurados en el espacio por medio de una estructura tipo ***R-tree***, la búsqueda exhaustiva de los puntos en el conjunto de datos conllevará un tiempo máximo $O\left(\log n\right)$, con lo que la complejidad de nuestro algoritmo se vería relajada a $O\left(n \log n\right)$.

3.2.3. Experimentos y resultados.

Si aplicamos este algoritmo al conjunto de datos mostrado en la figura 3.1a, obtenemos el resultado mostrado en la figura 3.3. Como vemos, este algoritmo realiza correctamente el agrupamiento, mientras que K-Means no lo resuelve, tal como se comprueba en la figura 3.1b.

La mayoría de los algoritmos no son capaces de detectar la presencia de ruido en el conjunto de datos. Si aplicamos el algoritmo K-Means al conjunto de datos representado en la figura 3.4 (en la que se pueden apreciar claramente dos grupos rodeados de ruido), los grupos que se forman no son los deseados y el ruido no es detectado, tal como se observa en la figura 3.5. En cambio, si lo aplicamos al algoritmo DBSCAN sí se obtiene el resultado deseado; se forman los grupos que buscamos y se detecta ruido. Si se observa la figura 3.6, se puede apreciar como el algoritmo detecta dos grupos, representados en colores azul y naranja, pero además detecta el ruido, representado en la figura como aquellos puntos mostrados como cuadrados azules.

Algorithm 5 Algoritmo DBSCAN

Entrada: Conjunto de datos $D \in \mathbb{R}^d$, con n puntos, radio Eps y $MinPts$;

while $\exists\, p \in D$ sin clasificar **do**
  coger $p \in D$;
  if p sin clasificar **then**
    $N_{Eps}(p) \leftarrow \{q \in D | dist(p,q) \leq Eps\}$;
    if $|N_{Eps}(p)| > MinPts$ **then**
      $type(p) \leftarrow core$;
      $class(p) \leftarrow clusId$;
    end if
    if p es core **then**
      crear lista $\Phi = \{x \in D | x \in N_{Eps}(p)\}$;
      while Φ tiene objetos **do**
        coger $o \in N_{Eps}(p)$;
        $N_{Eps}(o) \leftarrow \{q \in D | dist(o,q) \leq Eps\}$;
        if $|N_{Eps}(o)| > MinPts$ **then**
          $type(o) \leftarrow core$;
          for all objecto $i \in N_{Eps}(o)$ **do**
            if i esta $sin\ clasificar$ o es $ruido$ **then**
              if i esta $sin\ clasificar$ **then**
                añadir objeto i a Φ;
              end if
              if i es $ruido$ **then**
                $type(i) \leftarrow border$;
              end if
              $class(i) \leftarrow clustId$
            end if
          end for
        else
          $type(o) \leftarrow border$;
        end if
        $class(o) \leftarrow clusId$;
        eliminar o de Φ;
      end while
    else
      $type(p) \leftarrow noise$;
      $class(p) \leftarrow noise$;
    end if
  end if
  $clustId \leftarrow clusId + 1$;
end while

Salida: vector $class$ y vector $type$.

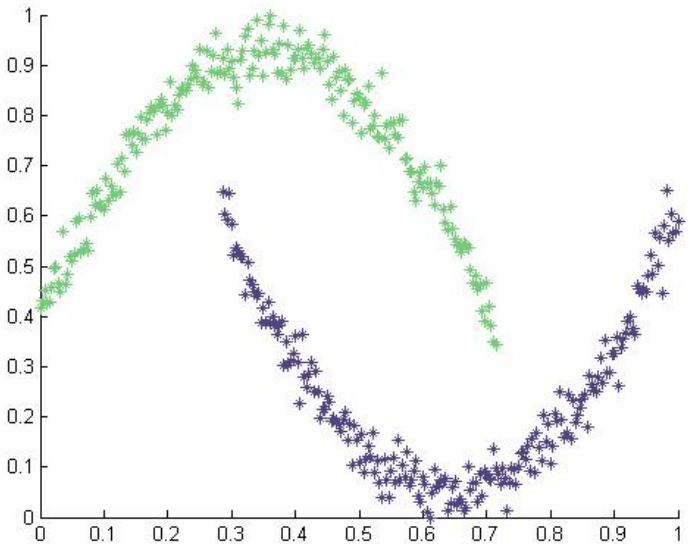


Figura 3.3: Resultado de aplicar el algoritmo DBSCAN al conjunto de datos 1, para $Eps = 0{,}05$ y $MinPts = 4$.

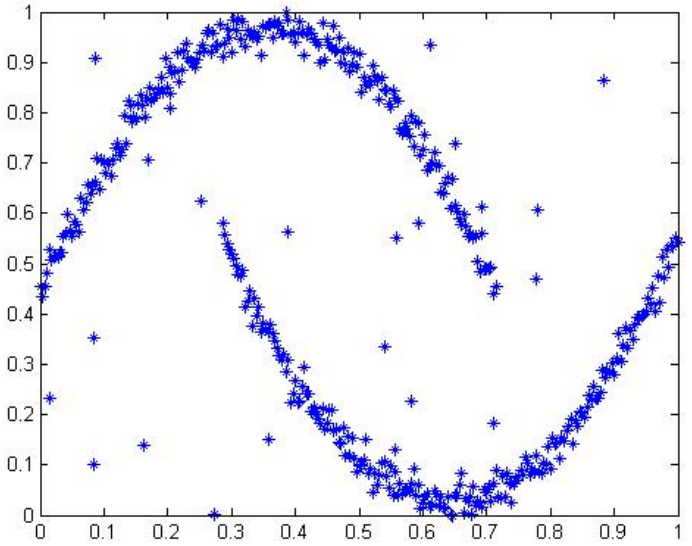


Figura 3.4: Conjunto de datos 2.

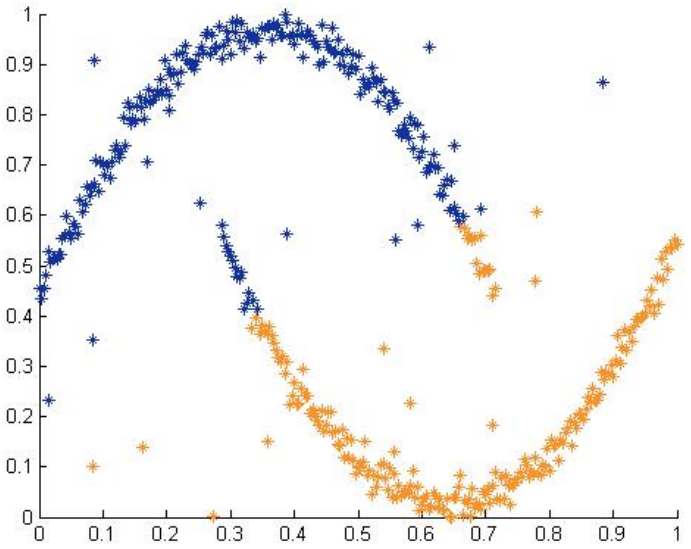


Figura 3.5: Resultado de aplicar el algoritmo K-Means al conjunto de datos 2, para un número de grupos igual a 2.

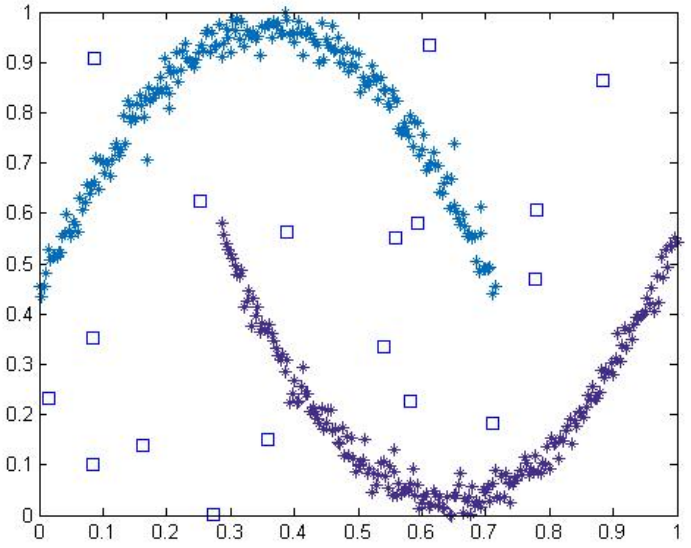


Figura 3.6: Resultado de aplicar el algoritmo DBSCAN al conjunto de datos 2, para $Eps = 0{,}05$ y $MinPts = 4$.

Como se explicó previamente, el algoritmo DBSCAN define tres tipos de puntos o elementos: centrales, frontera y ruido. Si se aplica a un conjunto de datos como el mostrado en la figura 3.7 el algoritmo DBSCAN obtenemos un resultado como el de la figura 3.8. En esta figura se representan los dos grupos a través de dos colores, el color lila representa un grupo y el azul otro. Además, en esta figura se representan los tres tipos de puntos. Los puntos que son ruido están representados por medio de cuadrados azules, los puntos frontera se representan con un círculo y los puntos centrales figuran como asteriscos.

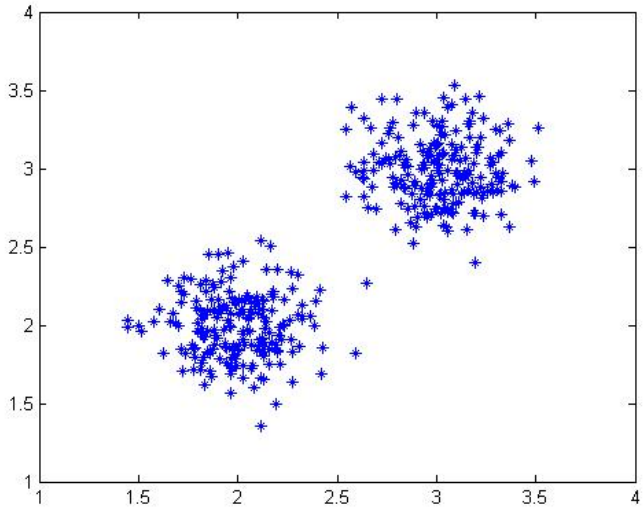


Figura 3.7: Conjunto de datos 3.

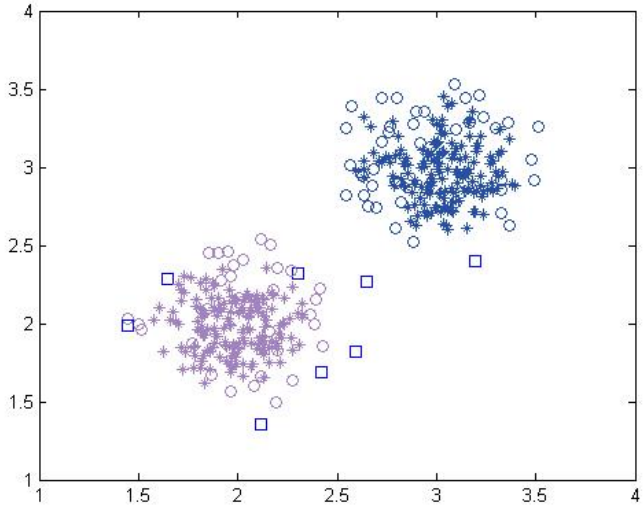


Figura 3.8: Tipos de puntos obtenidos al aplicar el algoritmo DBSCAN al conjunto de datos 3, para $Eps = 0{,}2$ y $MinPts = 5$.

3.2.4. DBSCAN 4C.

La principal razón por la que son útiles los algoritmos de agrupamiento es porque a través de ellos conocemos la estructura de los grupos formados, lo cual resulta de especial interés ya que distintos grupos representan con frecuencia a distintos tipos de objetos, distinción que antes era desconocida o inapreciable. De este modo, los grupos ofrecen otra visión del conjunto de datos que puede ser explotada de otra manera.

Otro tipo de información que puede estar oculta entre los datos y que puede aportar más información relevante al usuario, son las correlaciones presentes en el conjunto de datos. Una correlación es una dependencia lineal entre una o más características del conjunto de datos. El método más conocido para detectar correlaciones lineales es el análisis de componentes principales o PCA (*principal component analysis*) [54] pero presenta el inconveniente de que sólo se puede aplicar para el conjunto entero de datos, de modo que sólo se pueden encontrar correlaciones que están presentes en todo el conjunto de datos.

En el trabajo presentado por C. Böhm et al. [46] se presenta un método, DBSCAN-4C, que es capaz de encontrar subconjuntos de datos que contienen fuertes correlaciones y los cuales están densamente poblados. Estos subconjuntos se denominan grupos conectados y correlados (*correlation connected cluster*). Este método une los conceptos desarrollados en el algoritmo DBSCAN [2] y el de PCA [54] para encontrar este tipo de agrupamientos correlados.

En los siguientes subapartados se presentan las bases fundamentales para la formulación del algoritmo DBSCAN-4C y su implementación, así como algunos resultados obtenidos.

3.2.4.1. Fundamentos.

Tal como se ha explicado anteriormente, el objetivo final de este algoritmo es encontrar grupos conectados y correlados. Estos grupos son regiones densas de puntos en el espacio de características de dimensión N, que contienen, al menos, un eje principal en el cual los datos tienen una variación mínima. De este modo los grupos conectados y correlados poseen dos propiedades: son grupos densamente conectados y son grupos cuyos datos están correlados.

En el apartado 3.2.1 se estudiaron todos los aspectos necesarios para formular una definición de un grupo densamente conectado (*density connected cluster*). No obstante, para identificar los grupos conectados y correlados es necesario saber qué son conjuntos correlados. El análisis de componentes principales determina la matriz de covarianza del conjunto de datos S como: $\mathbf{M} = [m_{ij}]$ donde $m_{ij} = \sum_{s \in S} s_i s_j$; después descompone esta matriz, **M**, en una matriz ortonormal, **V**, denominada matriz de vectores propios, y en una matriz diagonal, **E**, denominada matriz de valores propios. Un conjunto forma un hiperplano correlado de dimensión λ, si existen $d - \lambda$ valores propios que tienen un valor menor que un determinado umbral, $\delta \approx 0$. De este modo se puede definir un conjunto linealmente correlado de dimensión λ (λ-*dimensional linear correlation set*) como [46]:

Definición 6 Sea un subconjunto de datos, $S \subseteq D$, $\lambda \in \mathbb{N}\,(\lambda \leq d)$, $\mathbf{E} = e_1, \ldots, e_d$ los valores propios de S en orden descendiente (esto es: $e_i \geq e_{i+1}$) y $\delta \in \mathbb{R}\,(\delta \approx 0)$. S forma un conjunto linealmente correlado de dimensión λ con respecto a δ, si por lo menos existen $d - \lambda$ valores propios en S que son próximos a cero. Esto es:

$$CorSet_{\delta}^{\lambda}\,(S) \Leftrightarrow |\{e_i \in \mathbf{E} | \Omega\,(e_i) \leq \delta\}| \geq \mathrm{d} - \lambda$$

siendo $\Omega\,(e_i) = e_i / e_{max}$ una normalización de los valores propios de $[0, 1]$.

De igual manera, se puede definir una *dimensión de correlación* de un subconjunto $S \in D$, $CorDim\,(S)$, como el número de valores propios con $e_i > \delta$. De este modo, si un subconjunto S es un conjunto linealmente correlado de dimensión λ, entonces, $CorDim\,(S) \leq \lambda$ [46].

Para formalizar el concepto de conjuntos conectados y correlados es necesario unir los conceptos de conjuntos densamente conectados (definidos en el algoritmo DBSCAN, definición 4) y de conjuntos correlados (definición 6). La idea fundamental es la de encontrar puntos centrales en un grupo, de tal manera que estos puntos tengan una determinada dimensión de correlación y un número mínimo de puntos en su región de vecindad dado el radio Eps. De este modo, asociamos cada punto, P, a la matriz de similitud o de covarianzas, $\hat{\mathbf{M}}_P$ calculada a través de la región de vecindad. Un punto P se incluye en el mismo grupo, si tiene una matriz de covarianzas parecida o igual que la de otros puntos que pertenecen a ese grupo. Para conseguir esto, el algoritmo busca puntos que están próximos al eje principal de aquellos puntos que pertenecen al grupo. Con estos datos se define una nueva matriz de covarianzas $\hat{\mathbf{M}}_P = \mathbf{V}_P \hat{\mathbf{E}}_P \mathbf{V}_P^T$, donde $\hat{\mathbf{E}}_P = \{\hat{e}_1, \ldots, \hat{e}_d\}$ es la nueva matriz de vectores propios calculada a partir de la matriz $\mathbf{E}_P = \{e_1, \ldots . e_d\}$, teniendo en cuenta la regla:

$$\hat{e}_i = \begin{cases} 1 & \text{if } \Omega\,(e_i) > \delta \\ 50 & \text{if } \Omega\,(e_i) \leq \delta \end{cases}$$

Al aplicar esta regla, se consigue que los puntos en la dirección de máxima variación tengan mayor relevancia, mientras que aquellos que están en otras direcciones tienen un peso menor.

La medida de disimilitud entre dos puntos del conjunto de datos, P y Q, usada en este algoritmo es: $dist_P\,(P, Q) = \sqrt{(P - Q)\,\hat{\mathbf{M}}_P\,(P - Q)^T}$. Hemos de tener en cuenta que esta medida no es simétrica, es decir, no siempre se cumple $dist_P\,(P, Q) = dist_Q\,(Q, P)$, por ello para calcular la medida de similitud entre dos puntos se calcula el máximo de las dos distancias anteriores, esto es: $\max\,(dist_P\,(P, Q)\,, dist_Q\,(Q, P))$. Con esta restricción no se pueden usar las definiciones de región de vecindad, o de punto central según se vio en DBSCAN. Las siguientes definiciones son necesarias para formalizar el concepto de grupos conectados y correlados [46]:

Definición 7 Sea $Eps \in \mathbb{R}$, se define la *región de vecindad correlada* dado un radio Eps de un punto $P \in D$, denominada $N_{Eps}^{\hat{\mathbf{M}}_P}\,(P)$, como:

$$N_{Eps}^{\hat{\mathbf{M}}_P}\,(P) = \{X \in D | \max\,(dist_P\,(P, Q)\,, dist_Q\,(Q, P)) \leq Eps\}$$

Definición 8 Sea $Eps, \delta \in \mathbb{R}$ y $MinPts, \lambda \in \mathbb{N}$, un punto $P \in D$ se define como *punto central correlado* con respecto a Eps, δ, $MinPts$, λ (denominado: $Core_{den}^{cor}(P)$), si su región de vecindad dado el radio Eps es un conjunto linealmente correlado de dimensión λ, y su región de vecindad correlada dado el radio Eps contiene un mínimo de puntos $MinPts$, es decir:

$$Core_{den}^{cor}(P) \Leftrightarrow CorSet_{\delta}^{\lambda}(N_{Eps}(P)) \wedge \left| N_{Eps}^{\hat{\mathbf{M}}_P}(P) \right| \geq MinPts$$

donde «*cor*» se refiere a los parámetros de correlación, δ y λ, y «*den*» a los parámetros de densidad, Eps y $MinPts$.

Definición 9 Sea $Eps, \delta \in \mathbb{R}$ y $MinPts, \lambda \in \mathbb{N}$, un punto $P \in D$ es *directamente alcanzable y correlado* desde otro punto $Q \in D$ con respecto a Eps, δ, $MinPts$, λ (denominado: $DirReach_{den}^{cor}(Q, P)$), si Q es un punto central correlado, la dimensión de correlación de $N_{Eps}(P)$ es por lo menos λ y $P \in N_{Eps}^{\hat{\mathbf{M}}_Q}(Q)$, es decir:

$$DirReach_{den}^{cor}(Q, P) \Leftrightarrow$$

$$1.\ Core_{den}^{cor}(Q)$$

$$2.\ CorDim(N_{Eps}(P)) \leq \lambda$$

$$3.\ P \in N_{Eps}^{\hat{\mathbf{M}}_Q}(Q)$$

Definición 10 Sea $Eps, \delta \in \mathbb{R}$ y $MinPts, \lambda \in \mathbb{N}$, un punto $P \in D$ se define como *alcanzable y correlado* desde un punto $Q \in D$ con respecto a Eps, δ, $MinPts$, λ ($Reach_{den}^{cor}(Q, P)$), si existe una cadena de puntos $P_1, \ldots, P_n$, tal que $P_1 = Q$,$P_n = P$ y P_{i+1} es directamente alcanzable y correlado desde P_i. Es decir:

$$Reach_{den}^{cor}(Q, P) \Leftrightarrow$$
$$\exists P_1, \ldots, P_n \in D : P_1 = Q \wedge P_n = P \wedge \forall i \in \{1, \ldots, n-1\} : DirReach_{den}^{cor}(P_i, P_{i+1})$$

Definición 11 Sea $Eps \in \mathbb{R}$ y $MinPts \in \mathbb{N}$, un punto $P \in D$ se define como *conectado y correlado* a un punto $Q \in D$, si hay un punto $O \in D$, tal que ambos, P y Q son alcanzables y correlados desde O. Es decir:

$$Connect_{den}^{cor}(Q, P) \Leftrightarrow \exists O \in D : Reach_{den}^{cor}(O, P) \wedge Reach_{den}^{cor}(O, Q).$$

La relación «conexión-correlación» es una relación simétrica. Con ella, se puede formalizar una definición de un conjunto o grupo conectado y correlado (*correlation connected cluster*):

Definición 12 Sea $Eps, \delta \in \mathbb{R}$ y $MinPts, \lambda \in \mathbb{N}$, un subconjunto no vacio $C \subseteq D$ se denomina *grupo conectado y correlado* con respecto a Eps, δ, $MinPts$, λ, si todos los puntos en C estan conectados y correlados, y si es máximo con respecto a la alcanzabilidad y a la correlación. Es decir:

$$ConnSet_{den}^{cor}(Q, P) \Leftrightarrow$$
$$(1)\ \forall O, Q \in C : Connect_{den}^{cor}(O, Q)$$
$$(2)\ \forall P, Q \in C : Q \in C \wedge Reach_{den}^{cor}(Q, P) \Rightarrow P \in C.$$

Al igual que en DBSCAN, tal como se explica en [46], los autores también definen dos lemas para validar la corrección del algoritmo implementado. En general, estos dos lemas indican que se puede encontrar un conjunto conectado y correlado siguiendo dos pasos: primero, se escoge aleatoriamente, un objeto central correlado $O \in D$; y segundo, se recogen todos los puntos que sean alcanzables y correlados desde O. Este proceso, lleva a la formación de un grupo densamente conectado y correlado que contiene a O.

3.2.4.2. Algoritmo.

Una vez presentados los principios básicos necesarios para la formulación del algoritmo, podemos explicar su implementación. La idea principal consiste en que al principio todos los objetos estén sin clasificar y a medida que el algoritmo se vaya ejecutando los puntos sean clasificados en algún grupo o definidos como ruido. Primero, se escoge un punto sin clasificar al azar; después, se comprueba si este es un punto central correlado. Si es así, se buscan los puntos que son densamente alcanzables y correlados desde éste y se comienza a expandir el grupo. Si no es así, el punto se clasifica como ruido. Al igual que para el algoritmo DBSCAN, un punto que es clasificado inicialmente como ruido y posteriormente, en algún punto de la ejecución, se demuestra que es alcanzable desde otro punto, entonces es clasificado dentro del grupo al que pertenece el punto desde el que es alcanzable y se clasifica como punto frontera.

El proceso de expansión de un grupo comienza al crearse una lista con todos los puntos que son densamente alcanzables y correlados desde el punto central y correlado inicialmente escogido, es decir, aquellos que pertenecen a la región de vecindad correlada de este punto. Después, se escoge un punto de la lista, y es clasificado como frontera o central y correlado según corresponda. Si el punto es central y correlado, se añaden a la lista todos los puntos que se encuentran en la región de vecindad de este nuevo punto. Cada vez que un punto de esta lista es clasificado, se borra de ella y se continúa por el siguiente punto de la lista siguiendo los mismos pasos. Así sucesivamente, hasta que la lista este vacía. Cuando la lista este vacía se busca aleatoriamente otro punto del conjunto de datos que sea central y correlado y que esté sin clasificar, y se comienza a expandir un nuevo grupo del mismo modo que se hizo con el punto anterior. El algoritmo 6 muestra el pseudocódigo con el proceso descrito anteriormente.

En este algoritmo se buscan los vecinos de cada punto del conjunto de objetos que sea central y correlado. Para comprobar que un punto es central y correlado, y buscar el número de vecinos se necesita un tiempo máximo de $O\left(d^2 \cdot n\right)$. Para esta búsqueda y esta comprobación es necesario calcular la matriz de covarianzas, lo cual se realiza en un tiempo máximo de $O\left(d \cdot n\right)$; después, esa matriz es descompuesta en sus vectores y valores propios, lo cual conlleva un tiempo máximo de $O\left(d^3\right)$. Si esto lo hacemos para cada punto, el algoritmo tardará un máximo de $O\left(d^2 \cdot n^2 + d^3 \cdot n\right)$. Si, del mismo modo que se hizo en DBSCAN, se ordenan los datos según una estructura que facilite su búsqueda, el tiempo de ejecución del algoritmo se puede reducir a $O\left(d^2 \cdot n \cdot \log n + d^3 \cdot n\right)$.

Algorithm 6 Algoritmo DBSCAN4C

Entrada: Conjunto de datos $D \in \mathbb{R}^d$, con n puntos, radio Eps, $MinPts$, λ y δ;

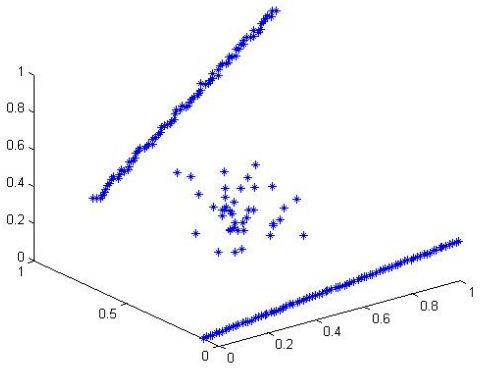
while $\exists\, p \in D$ sin clasificar **do**
  coger $p \in D$;
  if p sin clasificar **then**
    $N_{Eps}(p) \leftarrow \{q \in D | max()\, d_p(p,q), d_q(q,p) \leq Eps\}$;
    if $|N_{Eps}(p)| > MinPts$ **then**
      calcular $\mathbf{M}_p$
      if $CorDim(N_{Eps}(p)) \leq \lambda$ **then**
        calcular $\hat{\mathbf{M}}_p$ y $N_{Eps}^{\hat{\mathbf{M}}_p}(p)$
        if $|N_{Eps}^{\hat{\mathbf{M}}_p}(p)| \leq MinPts$ **then**
          $type(p) \leftarrow central\ y\ correlado$;
          $class(p) \leftarrow clusId$;
        end if
      end if
    end if
    if $type(p) = central\ y\ correlado$ **then**
      crear lista Φ con puntos en $N_{Eps}(p)$
      while Φ tiene objetos **do**
        coger $o \in \Phi$;
        $class(o) \leftarrow clusId$;
        calcular $R = \{i \in D | DirReach_{den}^{cor}(o,i)\}$;
        for all objeto $i \in R$ **do**
          if i esta $sin\ clasificar$ o es $ruido$ **then**
            if i esta $sin\ clasificar$ **then**
              añadir objeto i a Φ;
            end if
            if $type(i)$ es $ruido$ **then**
              $type(i) \leftarrow border$;
            end if
            $class(i) \leftarrow clustId$
          end if
        end for
        eliminar o de Φ;
      end while
    else
      $type(p) \leftarrow noise$;
      $class(p) \leftarrow noise$;
    end if
  end if
  $clustId \leftarrow clusId + 1$;
end while

Salida: vector $class$ y vector $type$.

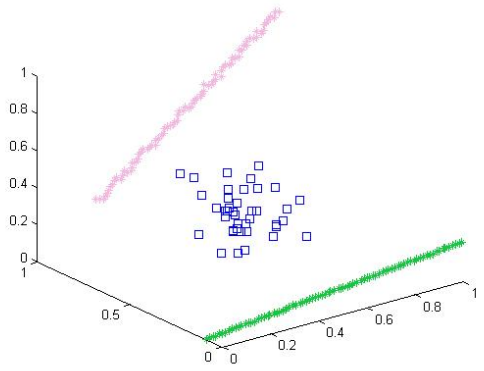
3.2.4.3. Experimentos y resultados.

La ventaja que presenta este algoritmo es que no sólo es capaz de encontrar la estructura del conjunto de datos a través de la búsqueda de grupos de puntos con una cierta densidad, sino que también es capaz de buscar relaciones entre estos puntos. Esto se puede observar si introducimos como conjunto de datos de entrada el representado en la figura 3.9a. Si como parámetros de entrada para el algoritmo usamos: $MinPts = 5$, $Eps = 7$, $\lambda = 2$ y $\delta = 0{,}01$ obtenemos el resultado de la figura 3.9b; donde observamos que el algoritmo descubre los grupos que tienen una dimensión de correlación igual a 2, es decir, que tienen una estructura definida en dos dimensiones.

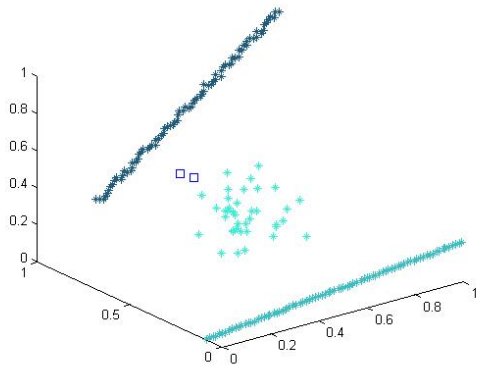
Si al mismo algoritmo le introducimos el mismo conjunto de datos y los mismos parámetros, exceptuando $\lambda = 2$ que es cambiado por $\lambda = 3$, vemos como en este caso el algoritmo encuentra grupos que tienen una dimensión de correlación igual a 3, es decir, que tienen una estructura definida en 3 dimensiones. Este resultado se puede visualizar en la figura 3.9c.



(a) Conjunto de datos 4.



(b) Resultado de DBSCAN4C, para $Eps = 0{,}15$, $MinPts = 4$, $\lambda = 2$ y $\delta = 0{,}001$.



(c) Resultado de DBSCAN4C, para $Eps = 0{,}15$ y $MinPts = 4$ $\lambda = 3$ y $\delta = 0{,}001$.

Figura 3.9: Conjunto de datos 4 y resultados de la ejecución del algoritmo DBSCAN 4C.

3.3. Algoritmo de agrupamiento Mean Shift.

En este apartado es presentada una técnica general no paramétrica para encontrar grupos en espacios de características. Este método fue desarrollado por D. Comaniciu y P. Meer en el año 2002 [47]. La utilización de esta técnica se basa en aplicar iterativamente el método para encontrar el máximo local o punto estacionario de la función de densidad más cercano a un punto P del conjunto de datos. La mayor ventaja que presenta este algoritmo es que no es necesario el conocimiento a priori del número de grupos y además permite la forma arbitraria de estos.

La razón para utilizar una técnica general no paramétrica de estimación de la densidad se debe a que el espacio de características se puede considerar como la función de densidad de probabilidad del parámetro representado. Regiones densas en el espacio de características se corresponden con los máximos locales de la función de densidad de probabilidad, esto es, las modas de la densidad desconocida. Una vez se ha encontrado la moda para cada punto, se asociarán en un grupo común a todas aquellas que se encuentren en un contorno determinado. Para encontrar cada una de las modas es necesario aplicar el algoritmo de Mean Shift que es explicado en el siguiente apartado.

Este algoritmo ha sido aplicado en diversos entornos. Un ejemplo es su aplicación en la segmentación y/o el filtrado de imágenes [47]. Otra de las aplicaciones más recientes y de mayor éxito ha sido en procesos de selección de características en alta dimensionalidad tal como se explica en [1].

3.3.1. Fundamentos.

Uno de los métodos más conocidos de estimación de la densidad es el basado en la estimación del kernel de densidad. Dado un conjunto de datos, $x_i, i = 1, \ldots, n$ de un espacio de dimensión d, $\mathbb{R}^d$, el *estimador del kernel de densidad multivariante* usando el Kernel $K(x)$ con radio de ventana o ancho de banda, h, es:

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x_i}}{h}\right).$$

Uno de los kernels más utilizados es el kernel gaussiano o normal, que es un kernel radialmente simétrico y se define como:

$$K_N(\mathbf{x}) = (2\pi)^{-d/2} exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)$$

Para todo Kernel radialmente simétrico, como es el caso del kernel normal, ha de cumplirse: $K(\mathbf{x}) = c_{k,d} k\left(\|\mathbf{x}\|^2\right)$, siendo $k(x)$ el perfil del kernel, para todo $x \geq 0$, y $c_{k,d}$ la constante de normalización que hace que la integral del kernel sea la unidad.

En el caso del kernel gaussiano o normal su perfil es:

$$k_N(x) = exp\left(-\frac{1}{2}x\right) \qquad x \geq 0$$

Pese a que en nuestro caso es utilizado un kernel gaussiano, se pueden usar otros kernels, como pueden ser el kernel uniforme o el Epanechnikov [47].

El primer paso en el análisis del espacio de características con densidad $f(\mathbf{x})$ es encontrar las modas de esta densidad. Estas modas se encuentran en los puntos donde el gradiente de la función de densidad se anula, es decir, cuando $\nabla f(\mathbf{x}) = 0$. El método Mean Shift es una manera elegante de encontrar estos puntos sin necesidad de estimar la función de densidad.

El gradiente del estimador de densidad se define como:

$$\begin{aligned}\hat{\nabla} f_{h,K}(\mathbf{x}) &= \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (\mathbf{x} - \mathbf{x_i})\, g\left(\left\|\frac{\mathbf{x} - \mathbf{x_i}}{h}\right\|^2\right) \\ &= \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x_i} - \mathbf{x}}{h}\right\|^2\right)\right] \left[\frac{\sum_{i=1}^{n} \mathbf{x_i} g\left(\left\|\frac{\mathbf{x}-\mathbf{x_i}}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x_i}}{h}\right\|^2\right)} - \mathbf{x}\right]\end{aligned}$$

donde $g(s) = -k'(s)$. El primer término es proporcional al estimador de densidad en $\mathbf{x}$ calculado a partir del kernel $G(\mathbf{x}) = c_{g,d} g\left(\|\mathbf{x}\|^2\right)$ y el segundo término:

$$\mathbf{m}_h(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{x_i} g\left(\left\|\frac{\mathbf{x}-\mathbf{x_i}}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x_i}}{h}\right\|^2\right)} - \mathbf{x} \tag{3.2}$$

es el *mean shift vector* o el vector de desplazamiento medio. Este vector apunta siempre a la dirección de máximo incremento de la densidad. El método Mean Shift se obtiene a través de la sucesión de los siguientes pasos:

- Cálculo del *mean shift vector* $\mathbf{m}_h\left(\mathbf{x}^t\right)$
- Translación de la ventana de desplazamiento $\mathbf{x}^{t+1} = \mathbf{x}^t + \mathbf{m}_h\left(\mathbf{x}^t\right)$

Este procedimiento garantiza la convergencia a un punto donde el gradiente de la función de densidad es cero, asegurando así la obtención de un punto estacionario de densidad máxima.

En el caso de utilizar un kernel gaussiano, la expresión del *mean shift vector* nos queda:

$$\mathbf{m}_h(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{x_i} exp^{-\frac{1}{2}\left\|\frac{\mathbf{x}-\mathbf{x_i}}{h}\right\|^2}}{\sum_{i=1}^{n} exp^{-\frac{1}{2}\left\|\frac{\mathbf{x}-\mathbf{x_i}}{h}\right\|^2}} - \mathbf{x}$$

La ventaja de utilizar este kernel gaussiano es que el camino que se obtiene hasta llegar a la moda tiene una trayectoria suave, es decir, el ángulo entre dos *mean shift vectors* consecutivos es siempre menor que 90^o [47].

3.3.2. Algoritmo.

El algoritmo de agrupamiento basado en Mean Shift consiste en la aplicación práctica del procedimiento de búsqueda de modas, según se describe a continuación:

- Para los puntos del conjunto de datos dejar correr el procedimiento de búsqueda de puntos estacionarios de $\hat{f}_{K,h}(x)$ o modas.
- Deshacerse de los puntos intermedios reteniendo sólo el máximo local.

En otras palabras, el algoritmo Mean Shift agrupa un conjunto de datos de dimensión n asociando a cada punto con la moda o el pico de la función de densidad de probabilidad del conjunto de datos. Para cada punto, el algoritmo calcula la moda correspondiente y la asocia a él. En primer lugar, se define un kernel radial centrado en el punto y con ancho de banda h y se calcula la media de los puntos que caen dentro del área definida bajo el kernel. Después, el algoritmo desplaza la ventana o el kernel hacia la media y se va repitiendo el proceso hasta que converge, es decir, hasta que el *mean shift vector* calculado recursivamente se mantiene constante o hasta que su variación es inferior a un umbral determinado. En cada iteración, la ventana se desplazará hacia zonas de mayor densidad de puntos hasta que se alcance el pico, donde los datos están distribuidos de la misma manera en la ventana. El algoritmo 7 muestra el pseudocódigo del agrupamiento Mean Shift. Al finalizar el algoritmo, todos los puntos asociados a una moda de valor similar son asignados a un mismo grupo.

Algorithm 7 Algoritmo de agrupación Mean Shift

Entrada: Conjunto de datos X donde $x_i \in \mathbb{R}^d, i = 1 \ldots n$ y ancho de banda h.

1: **for all** $x_i \in X$ **do**
2: $x_t \leftarrow x_i$
3: **while** $m_h(x_t)\ != \ umbral$ **do**
4: Calcular $m_h(t)$;
5: $x_t \leftarrow x_t + m_h(t)$
6: **end while**
7: $ClustCent_i = \{x \in D | dist(x, x_t)$ es mínima$\}$
8: **if** $ClustCent_i$ no existe **then**
9: $Cluster_i =$ nuevo $ClusterId$;
10: **end if**
11: **end for**

Salida: Vector $cluster \in \mathbb{R}^d$.

La figura 3.10 muestra el camino seguido hasta encontrar las modas de tres puntos. Los tres puntos de inicio, marcados con un cuadrado de color rojo, se encuentran rodeados por una ventana circular de radio 0.05, el mismo radio que se usa para el kernel gaussiano. Los cuadrados verdes describen cada uno de los desplazamientos necesarios hasta encontrar la moda, marcada con un cuadrado de color naranja. Se puede observar como los diferentes desplazamientos van siendo cada vez menores según se van acercando a la moda, debido a que el vector de

desplazamiento medio es cada vez menor al acercarse a las zonas de mayor densidad. También puede verse en la figura como dos puntos que inicialmente están en zonas diferentes, siguen trayectorias ascendentes en densidad hasta llegar a una moda común y por ello se considerarán miembros de un mismo grupo.

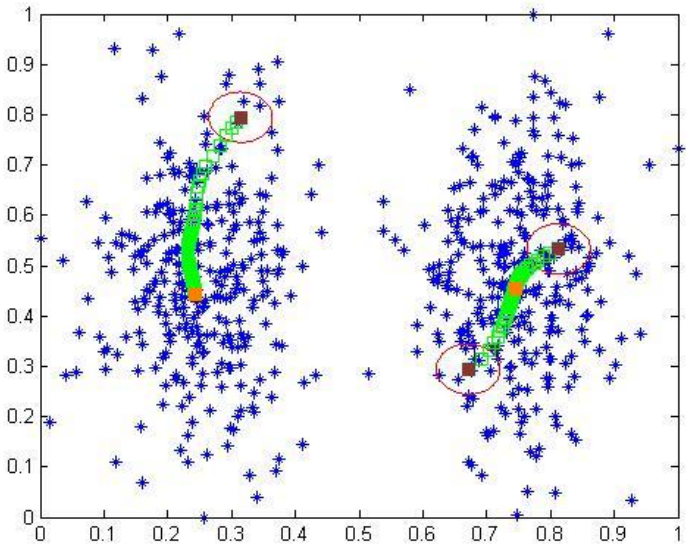


Figura 3.10: Proceso de búsqueda de modas del algoritmo Mean Shift para un ancho de banda $h = 0{,}05$.

Si aplicamos el conjunto de datos mostrado en la figura 3.1a, para $h = 0{,}05$, obtenemos un resultado óptimo, tal y como se puede observar en la figura 3.11

3.3.3. Mejoras aplicadas al algoritmo.

El algoritmo de agrupamiento Mean Shift tal como se ha presentado en el apartado anterior, presenta el inconveniente de ser demasiado lento para ciertas aplicaciones. Para solucionar este problema, se pueden incorporar algunas mejoras para acelerar el proceso de ejecución del algoritmo.

Básicamente, las mejoras que se han propuesto hasta ahora son dos. La primera se trata de la denominada «base de atracción». La idea consiste en que todos aquellos puntos cuya distancia a la moda es menor que el ancho de banda del kernel van a converger a esa misma moda. Al converger siempre a una misma moda, pueden ser asociados directamente a dicha moda sin tener así que repetir el proceso para cada uno de ellos.

La segunda mejora se basa en un principio similar al anterior. La idea es que aquellos puntos que se encuentran a una distancia menor que h/c del camino de convergencia hacia el pico, siendo c una constante y h el ancho de banda del kernel, estarán asociadas al mismo grupo

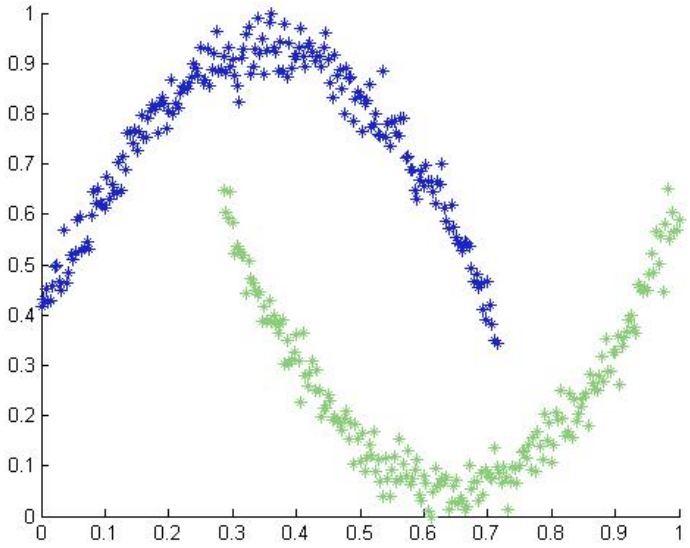


Figura 3.11: Resultado de aplicar Mean Shift para $h = 0{,}05$ al conjunto de datos 1.

que aquel que define la moda. En nuestro caso, escogemos $c = 4$ para asegurar la asignación correcta de los grupos y lograr una reducción de los tiempos de cálculo.

3.4. Curvas locales principales o *local principal curves* (LPC).

Los primeros en introducir el concepto de las curvas principales fueron T. Hastie y W. Stültzle [55]. Ellos definen las curvas principales como *curvas suaves de una dimensión que pasan por el centro de un conjunto de datos de dimensión p, ofreciendo un resumen no lineal de los datos*. A partir de entonces, este tipo de análisis ha tenido bastante seguimiento en la comunidad científica y han surgido en la literatura diversos algoritmos de búsqueda de curvas principales: Tibshirani [56], Kégl et al. (KKLZ, [57]), y recientemente Delicado [58]. La mayoría de estos algoritmos mantienen la estructura general, aunque difieren en la manera de encontrar el centro de la distribución.

En el año 2007, J. Einbeck, G. Tutz y L. Evers, propusieron en [4] un nuevo algoritmo de reconocimiento de curvas principales a partir de la búsqueda de centros de masa en los datos. Este algoritmo fusiona los conceptos de curvas principales usados por Delicado en [58] y de vector de desplazamiento medio usado por D. Comaniciu y P. Meers en [47]. Los centros de masa se encuentran a través de la definición de *mean shift vector* dada en el algoritmo anterior y el camino seguido por los centros definen la curva principal.

En el artículo desarrollado por J. Einbeck et al., en [4] el algoritmo es únicamente usado para obtener la curva local principal en un conjunto de datos. En nuestro trabajo, hemos desarrollado

un nuevo algoritmo que servirá no sólo para detectar la curva, sino también para agrupar todos los puntos pertenecientes a estas curvas y asignarlos a un mismo grupo. Este algoritmo también se basa en su esencia en la búsqueda de curvas locales principales descrito en el artículo de J. Einbeck et al. aunque se añade la fase de asignación.

Cada uno de los pasos que se deben seguir para encontrar los grupos definidos por las curvas locales principales se definen en el siguiente apartado. Estos pasos constituyen el algoritmo de agrupamiento basado en curvas locales principales.

3.4.1. Algoritmo.

Dado un conjunto de datos, $\mathbf{x}_i \in \mathbb{R}^d, i = 1, \ldots, n$, la curva que pasa por el centro de estos datos y, por tanto, el grupo asociado se puede obtener a través de la secuenciación de los siguientes pasos:

1. **Escoger el punto de inicio.**

   En principio, cualquier punto del conjunto de datos puede ser escogido para comenzar la búsqueda de las curvas principales. Existen dos posibilidades: escoger aleatoriamente el punto, o escoger el punto con mayor densidad $x_0 = \text{máx}_{x\in\mathbb{R}}\, \hat{f}\,(x)$, siendo $\hat{f}\,(x)$ el estimador del kernel de densidad multivariante. En nuestro caso, se escoge un punto de manera aleatoria para demostrar así la robustez del algoritmo pero, en cambio, el algoritmo ideado por J. Einbeck et al. opta por la segunda opción para asegurarse que no se comienza en un punto aislado.

2. **Asignación del grupo.**

   En el algoritmo de agrupamiento presentado es necesario asignar cada punto del conjunto de datos a un grupo. Cada grupo se corresponde con una curva.

   Como paso indispensable para la asignación de una curva a un grupo, es necesario que todos los puntos pertenecientes a la región de vecindad definida por el área dentro de la circunferencia de radio $h/2$ y de centro el punto $\mathbf{x}$, pertenezcan al mismo grupo al que pertenezca el punto x. Formalmente: $vecindad = \{x_i | dist\,(x, x_i) < h/2\}$

   En el algoritmo de búsqueda de curvas este paso no es necesario, ya que en él no se busca agrupar los datos del conjunto, sino encontrar las curvas existentes en este conjunto.

3. **Calcular el centro de masas.**

   El centro de masas de un conjunto de datos se obtiene a través de la siguiente ecuación:

$$\mu^x = \frac{\sum_{i=1}^{n} \mathbf{x}_i K\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)},$$

donde K es una función kernel de dimensión d. En esta implementación se opta por usar uno de los kernels más utilizados, este es, el kernel gaussiano o normal, que se define como:

$$K_N(\mathbf{x}) = (2\pi)^{-d/2} exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)$$

La ecuación del centro de masas esta íntimamente relacionada con la ecuación del vector de desplazamiento o *mean shift vector* 3.2 definido por D. Comaniciu y P. Meers en [47]. El vector de desplazamiento se obtiene como: $\mathbf{m}_h = \mu^x - \mathbf{x}$

4. **Calcular la componente principal.**

   Para calcular la componente principal es necesario definir la matriz de covarianza del punto x. Esta matriz se define como una matriz, Σ^x, de elementos $\sigma_{jk}, 1 \leq i, j \leq d$, obtenidos a partir de la ecuación:

$$\sigma^x_{jk} = \sum_{i=1}^{n} w_i \left(x_{ij} - \mu^x_j\right)\left(x_{ik} - \mu^x_k\right),$$

   donde w_i son los pesos definidos a través de la función kernel como:

$$w_i = \frac{K_N\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)}{\sum_{i=1}^{n} K_N\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)}.$$

   La componente principal, definida en este algoritmo como γ^x, se obtiene de escoger la primera columna de la matriz vectores propios resultante de la descomposición en valores y vectores propios de la matriz de covarianzas, Σ^x.

5. **Obtener el siguiente punto.**

   La recta definida por la componente local principal se puede obtener como $v^x(t) = \mu^x + t\gamma^x$, siendo t un valor real. El nuevo punto se obtiene como un punto dentro de esta recta: $x = \mu^x + t_0\gamma^x$, donde t_0 es igual al valor del ancho de banda, h.

   Previo a la generación del siguiente punto, se tiene que tener en cuenta que la dirección del vector propio puede variar de una iteración a otra. Para ello se comprobará que el coseno del ángulo formado entre el vector propio de la iteración i y el de la iteración $i-1$ es positivo o cero. Si éste es negativo, se cambiará γ^x_i por $-\gamma^x_i$. Este coseno se obtiene como $\cos(\alpha^x_i) = \gamma^x_{i-1} \circ \gamma^x_i$, donde α^x_i es el ángulo formado entre los vectores propios, y donde $\circ$ es el producto escalar.

6. **Repetir los pasos 2-5 hasta que μ^x se mantenga constante.**

   Cuando el margen en el conjunto de datos es alcanzado, el algoritmo se estanca y produce valores constantes de μ^x o valores donde la variación es pequeña o nula. En este momento se ha alcanzado el límite de la curva y se puede detener la ejecución.

7. **Repetir pasos 2-6 para la dirección opuesta.**

   Si el proceso comienza en un punto arbitrario de la curva, al alcanzar el límite de esta curva se termina el proceso. Pero puede que la curva no esté completamente definida. Es decir, si este punto no es un punto del límite opuesto de la curva, faltará por definir la otra mitad o parte de la curva. Para definir la otra parte de la curva, se realizarán los pasos del 2 al 5, pero teniendo en cuenta que ahora γ^x_i toma el valor de $-\gamma^x_i$.

Cuando este proceso haya terminado se habrá encontrado una curva completa, o lo que es lo mismo, un grupo. Si aún quedan puntos sin clasificar, todavía quedan nuevas curvas y grupos por descubrir. Por tanto, será necesario repetir los pasos 1-7 en los puntos sin clasificar. Este proceso se seguirá repitiendo hasta que todos los puntos se hayan clasificado, es decir, hasta que se hayan definido todas las curvas. El pseudocódigo del algoritmo de agrupamiento basado en *Local Principal Curves* se muestra a continuación:

Algorithm 8 Algoritmo de agrupamiento LPC

Entrada: Conjunto de datos X donde $\mathbf{x}_i \in \mathbb{R}^d, i = 1 \dots n$ y ancho de banda h.
Salida: $class(x)$ y $curves$.

for $i = 1$ hasta n **do**
  if x_i sin clasificar **then**
    $x \leftarrow x_i$
    repeat
      $class(ind_x) \leftarrow clusId$
      $curves(ind_x) \leftarrow x$
      $ind \leftarrow Find\left(x_i | dist(x, x_i) < h/2\right)$
      $class(ind) \leftarrow clusId$
      $\mu^x = \frac{\sum_{i=1}^{n} x_i K_N\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)}{\sum_{i=1}^{n} K_N\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)}$
      $\Sigma^x = \left(\sigma_{jk}^x\right)$
      $\sigma_{jk}^x = \sum_{i=1}^{n} w_i \left(x_{ij} - \mu_j^x\right)\left(x_{ik} - \mu_k^x\right)$, con $1 \leq j, d \leq d$
      donde $w_i = \frac{K_N\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)}{\sum_{i=1}^{n} K_N\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)}$
      $\Sigma^x = \Gamma^x \Lambda^x \left(\Gamma^x\right)^T$
      $\gamma^x \leftarrow$ primera columna de Γ^x
      $cos\left(\alpha_{(i)}^x\right) = \gamma_{i-1}^x \circ \gamma_i^x$
      if $cos\left(\alpha_{(i)}^x\right) < 0$ **then**
        $\gamma^x \leftarrow -\gamma^x$
      end if
      $x \leftarrow \mu^x + h\gamma^x$
    until μ^x se mantenga constante
    Repetir pasos 3-15 para dirección contraria $\gamma^x \leftarrow -\gamma^x$
  end if
  $clusId \leftarrow clusId + 1$
end for

La figura 3.12 muestra el proceso seguido hasta encontrar una curva local principal. En ella, el punto de inicio se encuentra marcado en color azul claro y está rodeado por una ventana circular de radio 0.05, mostrada con un círculo en rojo. En la figura, también se pueden observar, en color rojo, los puntos que se encuentran bajo este área y que son empleados para calcular la matriz de covarianzas, los valores y vectores propios, y posteriormente el desplazamiento necesario para crear un nuevo punto. Los puntos marcados como un cuadrado en verde describen cada

uno de los nuevos puntos calculados con el desplazamiento necesario y uniéndolos definen la curva local principal.

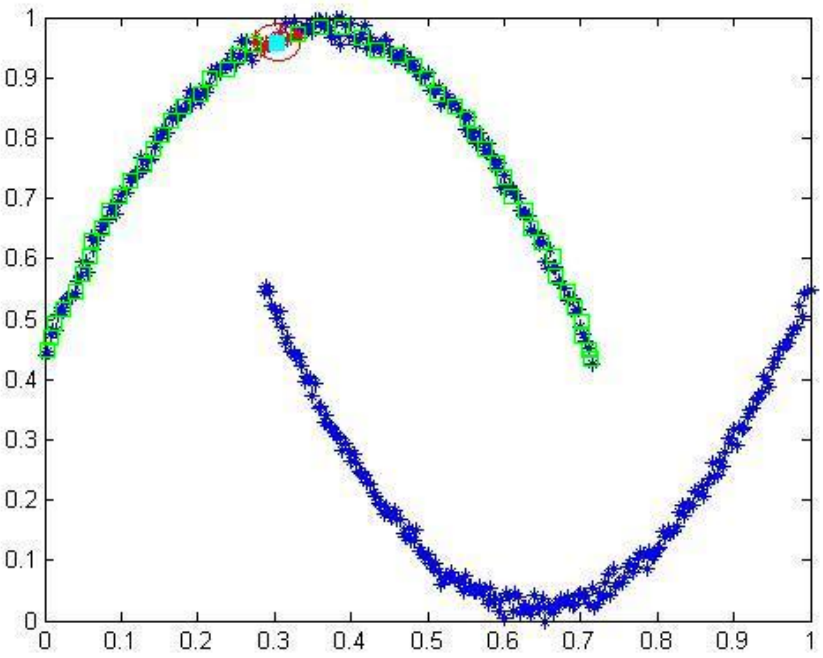


Figura 3.12: Proceso de búsqueda de una curva local principal usando el algoritmo LPC para un ancho de banda $h = 0{,}05$.

3.4.2. Experimentos y resultados.

Al igual que en los algoritmos anteriores vamos a aplicar el algoritmo LPC al conjunto de datos representados en la figura 3.1a, para verificar su funcionamiento correcto en la agrupación de formas curvas. El resultado obtenido se muestra en la figura 3.13, donde puede comprobarse visualmente que la agrupación es correcta.

La idea principal de este algoritmo es agrupar los elementos a través de la búsqueda de curvas principales. Si observamos la figura 3.14a, nos encontramos el conjunto de datos original en el cual existen dos grupos diferenciados, uno con forma de espiral y otro con forma cuadrática. En la figura 3.14b, comprobamos que el algoritmo LPC para agrupación ha encontrado los dos grupos, estando cada uno representado con colores distintos. Por otro lado, la figura 3.14c muestra los puntos que conforman las curvas suaves que pasan por el centro de cada uno de los grupos del conjunto de datos encontradas por el algoritmo. Si se unen todos estos puntos se obtienen las curvas locales principales. Como se observa en esta última figura, las curvas descubiertas son una buena aproximación de las curvas reales sin ruido.

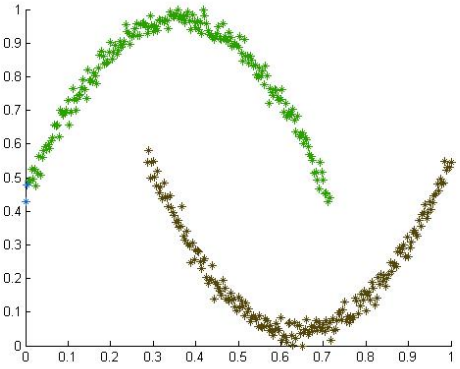
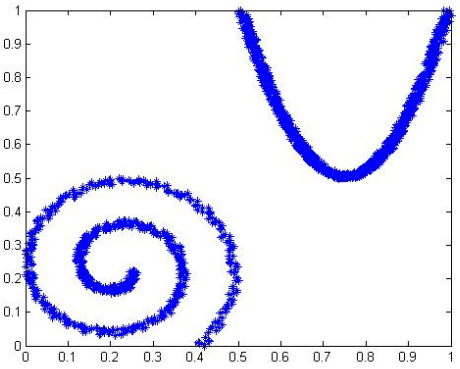
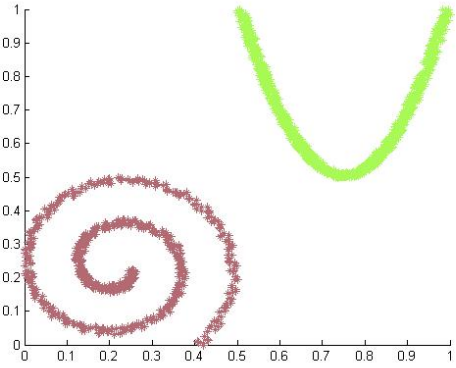


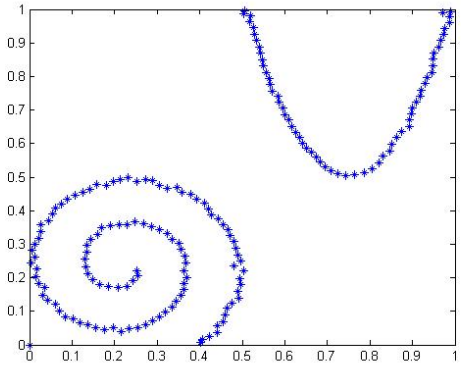
Figura 3.13: Resultado de aplicar LPC para $h = 0{,}05$ al conjunto de datos 1.



(a) Conjunto de datos 5.



(b) Grupos encontrados por LPC.



(c) Curvas encontradas por LPC.

Figura 3.14: Conjunto de datos 5 y resultado al aplicar LPC para $h = 0{,}02$.

Capítulo 4

Comparación de los algoritmos DBSCAN, Mean Shift Clustering y LPC.

En los apartados anteriores se ha definido en que consiste el agrupamiento y se estudiaron algunas de las técnicas o algoritmos de agrupamiento, haciendo especial hincapié en las técnicas de agrupamiento basadas en densidades. También se han explicado con detenimiento tres algoritmos de agrupamiento basados en densidades, DBSCAN, Mean Shift y LPC.

En este capítulo, comparamos estas tres técnicas de agrupamiento para, de este modo, mejorar su comprensión e interpretabilidad. De esta manera, se estudiarán las relaciones existentes entre los tres algoritmos y su comportamiento frente a las distintas características que son deseables en cualquier algoritmo de agrupamiento.

Los requisitos deseados por cualquier algoritmo de agrupamiento se pueden resumir entre otros en: la escalabilidad, la posibilidad de tratar con distintos tipos de atributos, la posibilidad de descubrir grupos con distinta forma, la capacidad de detectar ruido o *outliers*, la capacidad de manejar datos de gran dimensionalidad, la facilidad de interpretación o la posibilidad de tener un conocimiento mínimo del entorno para determinar los parámetros de entrada. El objetivo fundamental de este capítulo es demostrar si estos requisitos deseados se cumplen para los tres algoritmos previamente nombrados y a la vez hacer una comparación entre ellos.

Para alcanzar este objetivo, es necesario definir una serie de conjuntos de datos de pruebas, de modo que aplicados a estos algoritmos ofrezcan información relevante acerca de cada uno de ellos. En este capítulo, se analiza la respuesta de los algoritmos frente a datos que presentan ruido, a la variación de los parámetros de entrada, a la forma de los grupos que son capaces de detectar, al tiempo que tarda en ejecutarse, etc.

4.1. Conjunto de datos de pruebas.

El propósito principal de este trabajo es hacer un estudio comparativo de tres algoritmos de agrupamiento basados en densidades, estos son: DBSCAN, Mean Shift y LPC. En general, para poder hacer una buena comparación de cualquier algoritmo, es necesario definir una serie de conjuntos de datos de prueba a través de los cuales se puedan apreciar cada una de las ventajas e inconvenientes que presentan los algoritmos a comparar.

Un primer tipo de conjuntos de datos de prueba ha de ser creado de manera que se demuestre la independencia de los algoritmos a las formas de los grupos. Es decir, el conjunto se generará de modo que el resultado obtenido al ejecutar los algoritmos sobre él pueda demostrar que todos los algoritmos son capaces de detectar grupos de formas arbitrarias y localizar si existe alguna diferencia entre ellos. Para ello, se define un conjunto de datos que posee grupos con varias formas, un grupo en forma de espiral, otro lineal, otro gaussiano y, finalmente, otro cuadrático.

Por otro lado, al tratarse de algoritmos de agrupamiento basados en densidades, es muy importante saber cómo actúan estos algoritmos frente a conjuntos de datos de distintas densidades o conjuntos de datos que presentan densidades variables. Para ello, se genera un conjunto de datos en el que existen grupos con una densidad muy elevada en una parte central, y a medida que se alejan de este núcleo central tienen una densidad inferior. Esta cualidad se consigue fácilmente si utilizamos distribuciones gaussianas para generar los datos de los grupos. De igual manera, también se ha de observar qué sucede si en vez de tener grupos de densidades variables, los grupos tienen densidades constantes. Para ello, se genera un grupo de datos con forma lineal uniformemente distribuido.

Otro punto muy importante a tener en cuenta es la presencia de *outliers* y ruido en los conjuntos de datos. El estudio de cómo reaccionan los tres algoritmos frente a la presencia de ruido es de gran importancia a la hora de optar por uno u otro algoritmo. Para demostrar la respuesta de los algoritmos frente al ruido, se ejecutan los tres algoritmos para distintos conjuntos de datos con presencia de ruido y se estudia la tasa de acierto obtenida. El conjunto de datos de prueba se genera como dos grupos de forma cuadrática rodeados de ruido aleatorio de distribución normal.

La cantidad de datos que presentan los conjuntos es también un dato a tener en cuenta a la hora de realizar la comparación de estos algoritmos. Es muy probable que cada algoritmo reaccione de manera diferente ante estos atributos. Por ello, es necesario crear una base de datos que vaya incrementando en número de puntos.

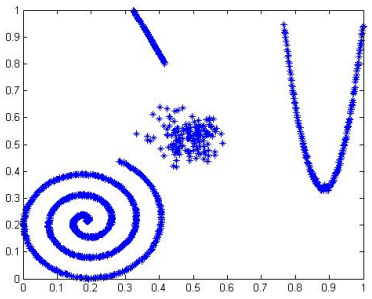
4.2. Efecto de la forma de los grupos sobre los resultados del algoritmo.

Una característica fundamental de los algoritmos basados en densidades es que, por lo general, son capaces de obtener buenos resultados para formas de los grupos arbitrarias. En este

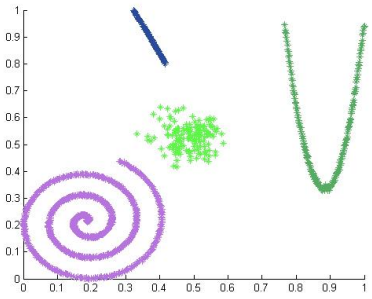
apartado, se pretende estudiar cómo reaccionan los tres algoritmos cuando son aplicados a un conjunto de datos con grupos de formas arbitrarias.

Para averiguar el efecto de la forma de los grupos, se ejecutan los tres algoritmos para el conjunto de datos mostrado en la figura 4.1a y, posteriormente, se observa el resultado obtenido para cada uno de los algoritmos. El conjunto de datos utilizado presenta cuatro grupos, uno en forma de espiral, otro lineal, uno gaussiano y uno cuadrático. Tal como se puede observar, este conjunto de datos no presenta ruido o *outliers*. Como parámetros de entrada se usa el mismo radio, 0.06, determinado heurísticamente, para el cual los tres algoritmos obtienen el mejor resultado y, además, para DBSCAN imponemos que el número mínimo de puntos de la región de vecindad es 4.

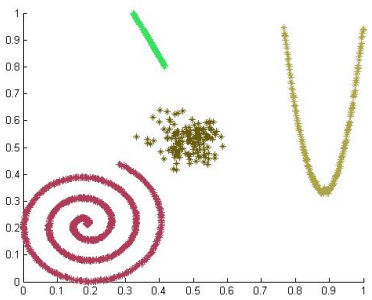
En las figuras 4.1b y 4.1c se puede observar que tanto el algoritmo DBSCAN como Mean Shift funcionan correctamente e identifican los grupos sin ningún problema. En cambio, como se observa en la figura 4.1d, el algoritmo LPC no es capaz de encontrar uno de los grupos. Esto sucede puesto que este grupo sigue una distribución gaussiana donde los puntos están muy dispersos y este algoritmo sólo reconoce grupos con forma de curva.



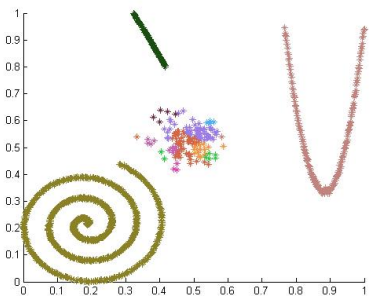
(a) Conjunto de datos 6



(b) Resultado de aplicar DBSCAN



(c) Resultado de aplicar Mean Shift



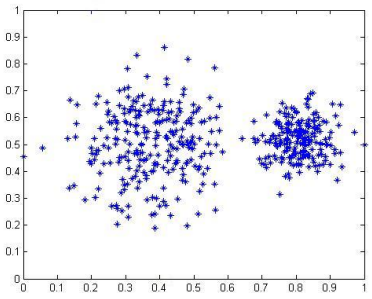
(d) Resultado de aplicar LPC

Figura 4.1: Resultados de la ejecución de los algoritmos al aplicar el conjunto de datos 6.

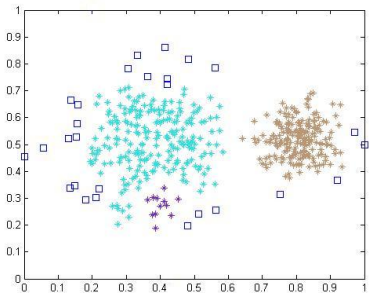
4.3. Efecto de la densidad de los grupos sobre los resultados del algoritmo.

No sólo la forma de los grupos es importante a la hora de estudiar el comportamiento de estos algoritmos. Es importante tener en cuenta que en este tipo de algoritmos basados en densidades la distribución de los puntos tiene un papel crucial. Es decir, se ha de tener en cuenta que las densidades de los grupos influyen sobre los resultados de los algoritmos.

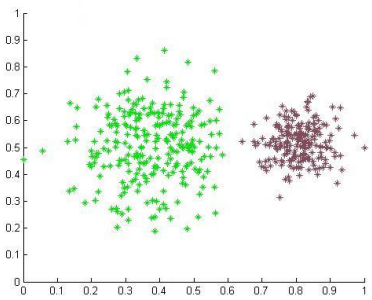
Si se observa el conjunto de datos de la figura 4.2a vemos que los puntos siguen una distribución gaussiana y que la densidad va variando a lo largo del grupo, siendo muy densa en el centro y presentando zonas de menor densidad en los bordes, pero sin llegar a considerarse ruido. Si aplicamos los tres algoritmos cuestionados a este conjunto de datos, obtenemos los resultados mostrados en las figuras 4.2b, 4.2c, 4.2d. Se puede comprobar visualmente que el único algoritmo capaz de identificar correctamente los grupos es Mean Shift. DBSCAN, no actúa bien cuando las densidades varían, tal y como se observa en la figura 4.2b, ya que algunos puntos del borde son identificados como ruido y uno de los grupos aparece subdividido en dos grupos. El algoritmo LPC no es capaz de identificar ni la forma del grupo. Esto se debe a que LPC sólo es capaz de reconocer grupos con forma de curva, y no descubre grupos con puntos muy dispersos o con formas esféricas.



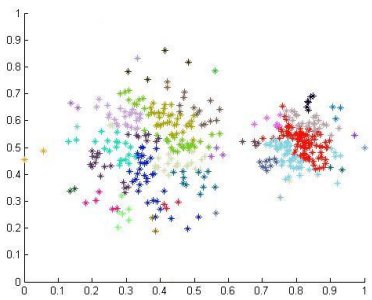
(a) Conjunto de datos 7



(b) Resultado de aplicar DBSCAN



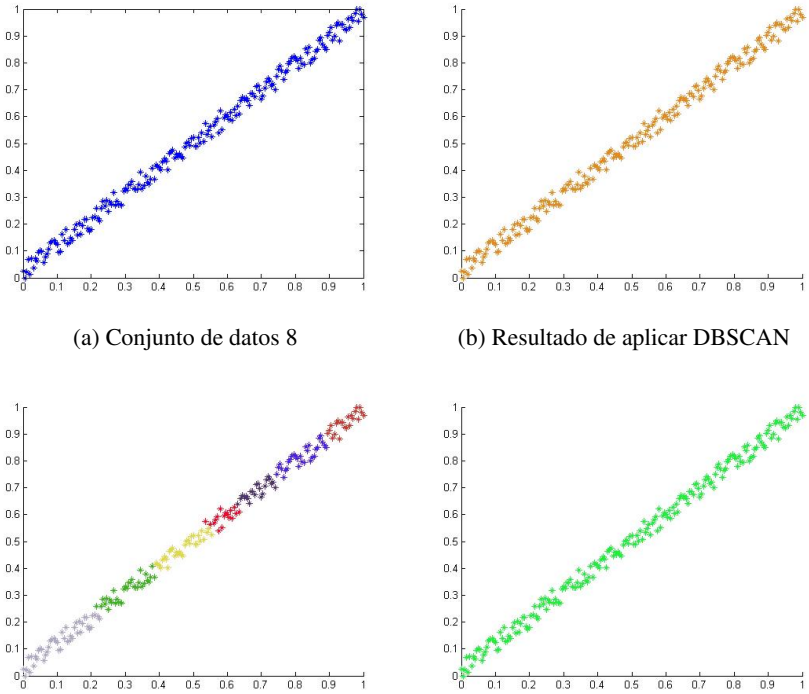
(c) Resultado de aplicar Mean Shift



(d) Resultado de aplicar LPC

Figura 4.2: Conjunto de datos 7 y resultados de la ejecución de los algoritmos.

Del mismo modo que las densidades variables a lo largo de los grupos pueden afectar a la efectividad del algoritmo de agrupamiento, grupos con densidades continuas también pueden afectar al agrupamiento. Para comprobar este hecho, se introduce un conjunto de datos como el mostrado en la figura 4.3a, donde existe un único grupo de forma lineal y con densidad constante. Las figuras 4.3b, 4.3c y 4.3d muestran los resultados obtenidos para estos tres algoritmos. Se puede comprobar que tanto DBSCAN como LPC son inmunes al hecho de que los grupos estén formados por densidades constantes. En cambio Mean Shift no reconoce el grupo, sino que divide el grupo en distintos subgrupos. Esto sucede, a causa de que el algoritmo Mean Shift reconoce los grupos asociando cada uno con el punto en la zona de mayor densidad. Es decir, supone que los grupos tienen una estructura con una parte más densa y otra menos densa, y cada punto del grupo es asociado con el centro de la zona de mayor densidad.



(a) Conjunto de datos 8

(b) Resultado de aplicar DBSCAN

(c) Resultado de aplicar Mean Shift

(d) Resultado de aplicar LPC

Figura 4.3: Conjunto de datos 8 y resultados de la ejecución de los algoritmos.

4.4. Capacidad de reconocer *outliers* o ruido.

El algoritmo DBSCAN presenta una ventaja sobre la mayoría de los algoritmos, y esta ventaja es la capacidad de reconocer la presencia de ruido o de reconocer *outliers* en los datos. *Mean Shift clustering* y *LPC clustering* no son capaces de reconocer ruido pero, se puede hacer una aproximación, se puede decir que los grupos encontrados cuyo número de puntos sea inferior a un cierto umbral serán clasificados como puntos de ruido.

Para estudiar la robustez de estos algoritmos frente al ruido, se realiza un estudio de la tasa de acierto obtenida en la clasificación a medida que aumenta el ruido en un conjunto de datos determinado. Para llevar a cabo este estudio, se utilizará el conjunto de datos definido previamente como conjunto 2 (mostrado en la figura 3.4), y realizando varios experimentos con un número de puntos de ruido uniformemente distribuido entre 0 y 1. Hay que puntualizar que el conjunto de datos 2 contiene 600 puntos sin ruido (cada grupo formado por 300 puntos).

La tasa de acierto será obtenida como el número de objetos clasificados correctamente (número de aciertos) dividido por el número de puntos del conjunto de datos. En esta clasificación se tendrán en cuenta para el cálculo de aciertos tanto los puntos de las curvas como los puntos de ruido. Así, si un punto de ruido es clasificado como parte de una curva, será considerado como error. De igual manera, si un punto de una curva es considerado como ruido, se considerará como error. La fórmula de la tasa de acierto se puede definir como:

$$Tasa_{acierto} = \frac{\text{Nº aciertos}}{\text{Nº de puntos totales}} \cdot 100$$

El experimento se realiza de la siguiente manera: se introduce el conjunto de datos con un número de puntos de ruido determinado a uno de los algoritmos y se obtiene la tasa de acierto. Posteriormente, se repite el algoritmo para el mismo conjunto de datos pero con más puntos de ruido que en el anterior, y se obtiene la nueva tasa de acierto, y así sucesivamente hasta que el número de puntos de ruido sea igual al número de puntos del conjunto de datos sin ruido. Debido a que la distribución del ruido en el espacio es aleatoria, se repite cada experimento $n = 10$ veces con una inicialización del ruido distinta (pero con el mismo número de puntos), logrando así una mayor exactitud en la medida de la tasa de acierto. Para cada n experimentos, se calculará la media de la tasa de acierto y su desviación típica como:

$$Tasa_{acierto\ media} = \frac{\sum_{i=1}^{n} Tasa_{acierto_i}}{n},$$

y la desviación típica como:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} \left(Tasa_{acierto\ media} - Tasa_{acierto_i}\right)^2}{n\,(n-1)}}.$$

Finalmente se calcula la tasa de acierto como:

$$Tasa_{acierto} = Tasa_{acierto\ media} \pm \sigma.$$

Así, para el algoritmo DBSCAN obtenemos de las tasas de acierto mostradas en la tabla 4.1 si se va incrementado el número de puntos de ruido en cada iteración. Para este experimento, se demuestra heurísticamente que el mínimo radio con el que obtenemos un agrupamiento correcto es para $Eps = 0{,}02$, teniendo en cuenta como número mínimo de puntos en la región de vecindad: $MinPts = 4$. El experimento se realizó añadiendo en cada iteración 40 puntos de ruido pero, en la tabla figura un resumen de este experimento.

Puntos de ruido	Tasa de acierto (%)
0	$100 \pm 0\,\%$
10	$96,774 \pm 4{,}737 \cdot 10^{-15}\,\%$
90	$76,923 \pm 0\,\%$
170	$63,83 \pm 0\,\%$
250	$54,545 \pm 0\,\%$
330	$51,714 \pm 0,084656\,\%$
410	$55,479 \pm 0,17115\,\%$
490	$57,873 \pm 0,19225\,\%$
570	$60,46 \pm 0,19003\,\%$

Tabla 4.1: Tasa de acierto para algoritmo DBSCAN aplicado al conjunto de datos 2 con distintos valores de ruido.

Si se utiliza la misma técnica para el algoritmo Mean Shift, habiendo demostrado de manera heurística que el menor ancho de banda para el cual se obtiene un resultado óptimo es $h = 0{,}05$, y definiendo como ruido todos aquellos puntos que son clasificados en un grupo que contiene un número de miembros inferior a 300 (ya que las dos curvas están compuestas de 300 puntos cada una), se obtiene las tasas de acierto mostradas en la tabla 4.2.

Puntos de ruido	Tasa de acierto (%)
0	$100 \pm 0\,\%$
10	$83,903 \pm 0,99071\,\%$
90	$68,308 \pm 1,8816\,\%$
170	$57,787 \pm 2,8197\,\%$
250	$46,345 \pm 1,6694\,\%$
330	$52,873 \pm 1,9295\,\%$
410	$55,845 \pm 1,5189\,\%$
490	$56,127 \pm 3,4869\,\%$
570	$56,069 \pm 2,1372\,\%$

Tabla 4.2: Tasa de acierto para algoritmo Mean Shift aplicado al conjunto de datos 2 con distintos valores de ruido.

Actuando del mismo modo que para Mean Shift, pero aplicando el algoritmo LPC con un ancho de banda mínimo $h = 0{,}02$ (demostrado de manera heurística), se consiguen los resultados de la tasa de acierto mostrados en la tabla 4.3.

Para facilitar la comprensión de los datos y comparar los resultados obtenidos, se muestran los datos de las tablas previas en la gráfica 4.4. En la gráfica se comprueba que, a medida que aumenta el ruido, disminuye la tasa de acierto hasta llegar a número de puntos de ruido igual al número de puntos de las curvas. En ese momento, aparece un punto de inflexión y la tasa de acierto comienza a aumentar. Esto se debe a la forma de calcular la tasa de acierto, ya que se incluyen como acierto los puntos de ruido clasificados como tal. De este modo, en un principio la tasa de acierto disminuye y cuando el número de puntos de ruido es igual al número de

Puntos de ruido	Tasa de acierto (%)
0	$100 \pm 0\,\%$
10	$94,71 \pm 2,2324\,\%$
90	$79,103 \pm 0,93178\,\%$
170	$62,83 \pm 1,8527\,\%$
250	$54,436 \pm 1,2389\,\%$
330	$51,698 \pm 1,8164\,\%$
410	$56,042 \pm 1,6091\,\%$
490	$58,215 \pm 2,5352\,\%$
570	$63,989 \pm 2,3224\,\%$

Tabla 4.3: Tasa de acierto para algoritmo LPC aplicado al conjunto de datos 2 con distintos valores de ruido.

puntos de los grupos se alcanza la entropía máxima y la tasa de acierto será igual al 50 %. Si a partir de este momento, se sigue aumentando el número de puntos de ruido, la tasa de acierto aumentará porque hay más puntos de ruido que son clasificados correctamente, siendo hipotéticamente posible llegar de nuevo a una tasa de acierto cercana al 100 %.

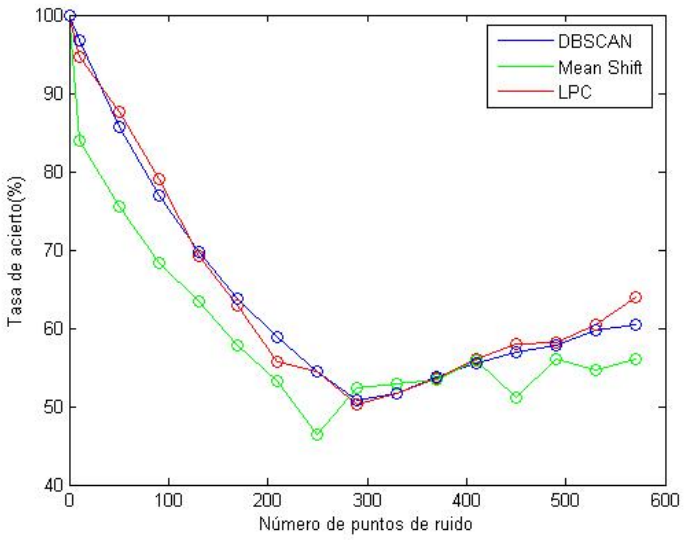


Figura 4.4: Tasa de acierto vs número de puntos de ruido.

Los tres algoritmos reaccionan de manera similar frente al ruido, teniendo en cuenta la suposición original, aunque Mean Shift presenta un comportamiento algo peor que los otros dos. La mayor diferencia estriba en la varianza. Se comprueba que DBSCAN tiene una varianza mínima o nula, lo cual hace que sea más robusto. LPC y Mean Shift son más sensibles a la inicialización aleatoria de los datos. Por tanto, podemos concluir que por robustez quedan ordenados de mayor a menor: DBSCAN, LPC y Mean Shift.

4.5. Tiempo vs Número de datos.

En este apartado se estudiará la escalabilidad de los algoritmos DBSCAN, Mean Shift y LPC. Se observará cómo aumentan los tiempos de cálculo de estos algoritmos cuando se incrementa el número de datos del conjunto de entrada.

El experimento realizado para estudiar el tiempo de ejecución de estos algoritmos frente al número de datos fue escoger un conjunto de datos como el conjunto de datos 1, con un total de 400 puntos, ejecutar los tres algoritmos y calcular el tiempo de ejecución de cada uno. Sucesivamente, se va repitiendo el proceso pero aumentado el conjunto de datos en 400 puntos hasta llegar a un total de 10000 puntos. En la tabla 4.4 se muestran algunos de los tiempos de ejecución obtenidos. El experimento se llevo a cabo en un ordenador marca Dell Pentium a 3 GHz con 1 Gb de memoria RAM.

Tiempo(s) / Nº de puntos	DBSCAN	Mean Shift	LPC
400	0,046	8,572	0,0001
1200	0,359	52,727	0,031
2000	0,734	123,36	0,032
2800	1,311	220,74	0,032
3600	2,092	369,19	0,047
4400	3,451	472,32	0,062
5200	4,185	698,18	0,062
6000	6,558	937,57	0,078
6800	8,368	1248,9	0,078
7600	10,211	1526,9	0,093
8400	11,46	1918	0,109
9200	14,892	2180,4	0,109
10000	15,078	2692,8	0,124

Tabla 4.4: Tiempo de ejecución vs número de datos.

Para facilitar la comparación, los resultados completos obtenidos en la ejecución de los tres algoritmos se muestran en la figura 4.5. En esta figura, el eje de ordenadas se representa en escala logarítmica, ya que la diferencia de tiempos entre los algoritmos es muy grande. Como podemos observar, el algoritmo que mejor actúa frente al tiempo es LPC seguido de DBSCAN. Mean Shift tarda mucho más tiempo en ejecutarse a medida que se incrementa el número de puntos y, por tanto, es el peor en escalabilidad.

4.6. Ventajas e inconvenientes.

Hasta ahora, se ha estudiado como se comportan los algoritmos DBSCAN, Mean Shift y LPC para conjuntos de entrada con grupos de formas arbitrarias y con densidades variables en los

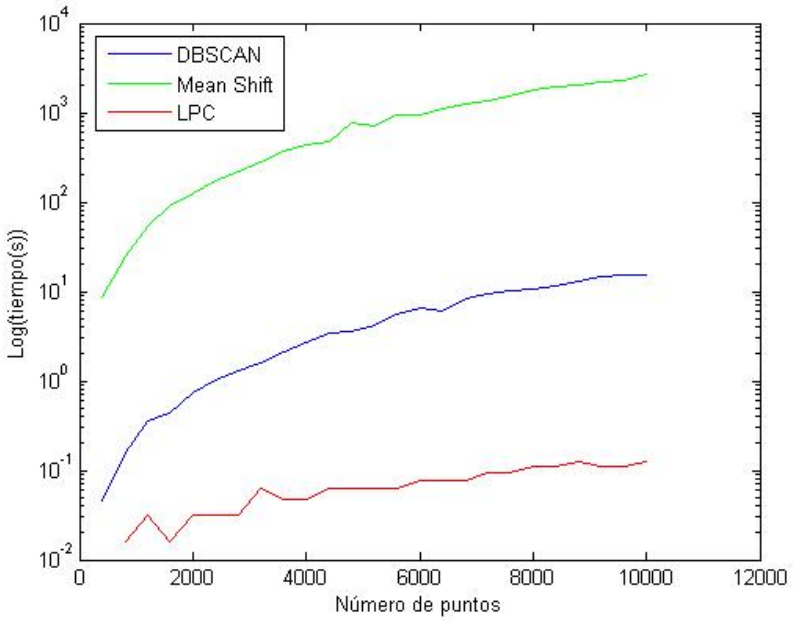


Figura 4.5: Tiempo de ejecución vs Número de datos.

grupos. En el segundo apartado de este capítulo se vio como DBSCAN y Mean Shift eran los más robustos a la hora de encontrar grupos de formas arbitrarias, y Mean Shift era el mejor de los tres cuando las densidades en los grupos no eran constantes.

Por otro lado, también se estudió cómo reaccionan los algoritmos frente al ruido. Por definición, el único algoritmo capaz de reconocer puntos de ruido es DBSCAN, luego se puede concluir que éste es el que mejor actúa ante la presencia de datos ruidosos. A pesar de esto, se ha realizado una aproximación para permitir la detección de ruido en las técnicas LPC y Mean Shift.

Per no sólo estas características son importantes a la hora de hacer una comparación. El número de parámetros de entrada y el conocimiento del entorno para determinarlos, la posibilidad de tratar distintos atributos, la escalabilidad, etc. también son características relevantes a la hora de escoger uno u otro algoritmo. A continuación, se explican las principales ventajas e inconvenientes que los algoritmos comparados presentan respecto a estas cualidades.

Parámetros de entrada. El número de parámetros de entrada también es importante, ya que son parámetros que se desconocen y que se deben suponer. Por lo tanto, cuantos menos parámetros existan menos asunciones serán necesarias, y generalmente más rápido será el método.

En algunos casos, el entorno nos ofrece algún tipo de información para poder determinar estos parámetros. En el caso de estos tres algoritmos, no existe ninguna forma de averiguar su valor

aproximado, aunque, para el algoritmo DBSCAN, se puede asumir que cuatro es un buen valor del número mínimo de puntos cuando el conjunto de datos es de dimensión dos, según se explica en [2]. Esto hace pensar que puede haber algún tipo de conocimiento heurístico aplicable para este método.

En el caso de DBSCAN el número de parámetros necesarios es dos. Por un lado, el número mínimo de puntos en un área de vecindad y, por otro lado, el radio del área de vecindad. En cambio, los algoritmos Mean Shift y LPC sólo necesitan un parámetro de entrada, es el radio o ancho de banda de la ventana de desplazamiento. Los tres casos presentan una ventaja sobre muchos otros algoritmos clásicos, y es el hecho de que no necesitan asumir el número de grupos, sino que son capaces de identificar los grupos existentes sin necesidad de este parámetro.

Posibilidad de tratar distintos tipos de atributos. Los conjuntos de datos se caracterizan por estar formados por atributos que pueden ser de distinto tipo. En general, nos encontramos con dos tipos de atributos: nominales y ordinarios (numéricos). Los datos nominales también pueden ser llamados categóricos, enumerados o discretos. Un requisito deseado en los algoritmos de agrupamiento es que sean capaces de tratar distintos tipos de atributos.

En el caso de los tres algoritmos estudiados, sólo se trata con datos numéricos pero, es posible hacer un preprocesado previo para que los algoritmos funcionen con distintos tipos de datos.

Robustez. Como se ha comentado en el apartado 4.4, el método más robusto frente al ruido y a las diferentes inicializaciones de éste, es DBSCAN. LPC y Mean Shift son menos robustos ya que su varianza frente al ruido es mayor. Además, por definición, ninguno de estos dos algoritmos detecta el ruido; para comprobar la robustez frente al ruido fue necesaria una suposición, según la cual, aquellos grupos cuyo número de miembros es inferior a un umbral son clasificados como ruido.

Observando la figura 4.4, podemos concluir que los tres algoritmos de agrupamiento no obtenemos buenos resultados frente al ruido, ya que la tasa de acierto no disminuye de manera lineal al aumentar el ruido.

Interpretabilidad El requisito de interpretabilidad mide la capacidad de una técnica para ayudar a la interpretación y comprensión de los datos que procesa. Todas las técnicas de agrupamiento ayudan a la interpretabilidad de los datos debido a que buscan estructuras homogéneas en conjuntos globales y faltos de estructura. Todos los algoritmos de densidad explicados en este proyecto, encuentran formas arbitrarias que proporcionan una interpretación espacial en vez de una interpretación puramente geométrica como los algoritmos tradicionales. Aunque los tres algoritmos estudiados poseen esta característica, se pueden ver algunas diferencias específicas de cada uno:

- DBSCAN es un algoritmo que obtiene los grupos conectando puntos que se encuentran en regiones densas del espacio.
- Mean Shift obtiene los grupos asociando cada punto a un pico o moda que se encuentra en la región de máxima densidad.
- LPC obtiene grupos con la característica de que están correlados.

La tabla 4.5 resume los requisitos que se cumplen para cada uno de los tres algoritmos.

	DBSCAN	Mean Shift	LPC
Escalabilidad	Media	Mala	Buena
Atributos	Numéricos	Numéricos	Numéricos
Parámetros de entrada	Eps y MinPts	Eps	Eps
Formas de los grupos identificados	Encuentra grupos de formas arbitrarias	Encuentra grupos de formas arbitrarias	No siempre encuentra grupos de formas arbitrarias
Efecto densidad de los grupos	Sensible a densidades variables	Sensible a densidad constante	Sensible a grupos de densidades muy dispersos
Ruido y *Outliers*	Reconoce ruido	No reconoce ruido	No reconoce ruido
Robustez	Media	Baja	Baja

Tabla 4.5: Comparación de los algoritmos DBSCAN, Mean Shift y LPC.

Capítulo 5

Conclusiones.

El presente Proyecto de Fin de Carrera está enmarcado dentro del área del aprendizaje automático y, más concretamente, en un conjunto de técnicas muy extendidas en este contexto, denominadas técnicas de agrupamiento (*clustering*). Por esta razón, en la primera parte del capítulo dos de esta memoria se han introducido brevemente los conceptos básicos de este campo, algunas de las técnicas de uso más comunes y sus aplicaciones más conocidas.

Las técnicas de agrupamiento han sido ampliamente utilizadas en campos tan diversos como la estadística, la informática o la medicina (p.e. *imaging*). Uno de los objetivos principales de este proyecto ha sido la presentación del estado del arte del agrupamiento o *clustering* dentro del área del aprendizaje automático. Este objetivo ha sido abordado en la segunda parte del capítulo dos, el cual está dedicado íntegramente a la exposición de los algoritmos de agrupamiento más comunes y a la descripción de su taxonomía atendiendo a sus características.

Una taxonomía clásica de las técnicas de agrupamiento es aquella que diferencia dos tipos de métodos: los jerárquicos y los particionales. Debido a la rápida expansión y los éxitos logrados dentro del área de minería de datos, ha habido un gran desarrollo de nuevos métodos y técnicas de agrupamiento. Esto ha provocado la ampliación de la taxonomía tradicional y la incorporación de dos nuevos tipos. Estos son los denominados métodos probabilísticos y los métodos basados en densidades. Los métodos basados en densidades rompen con las técnicas tradicionales de agrupamiento. Estas nuevas técnicas se basan en distancias espaciales con el fin de buscar formas arbitrarias teniendo en cuenta la densidad de puntos por área. Este tipo de agrupamiento surge dentro de la comunidad de aprendizaje automático (más concretamente en minería de datos) a mediados de los noventa [2] y, actualmente, sigue siendo un campo de investigación prolífico y con nuevas utilidades [1].

En el capítulo tres de esta memoria se ha profundizado en el estudio de este tipo de algoritmos y, en concreto, en estos tres: DBSCAN, Mean Shift y el agrupamiento LPC. Los algoritmos de agrupamiento basados en densidades surgieron para dar solución a problemas que los algoritmos clásicos no podían resolver. En estos algoritmos, los grupos son considerados como regiones densas en el espacio de datos que se encuentran separados entre sí por regiones de baja

densidad de objetos (ruido). Una característica de este tipo de algoritmos es que son capaces de encontrar grupos de formas arbitrarias que pueden estar distribuidos de cualquier manera. Algunas de las principales características de cada uno de estos tres algoritmos son:

- DBSCAN es un algoritmo de agrupamiento sencillo de implementar, basado en la búsqueda de puntos centrales en los conjuntos de datos. Estos puntos centrales conectan zonas de alta densidad de puntos hasta llegar a formar un grupo. Los puntos centrales se definen como puntos que poseen un área o región de vecindad para un radio determinado que contiene, al menos, un número mínimo de puntos.
- El agrupamiento Mean Shift aplica iterativamente la técnica no paramétrica de estimación de la densidad para encontrar el máximo local o punto estacionario de la función de densidad más cercano a un punto P del conjunto de datos. La razón para utilizar esta técnica se debe a que regiones densas en el espacio de características se corresponden con los máximos locales (modas) de la función de densidad de probabilidad. Una vez se haya encontrado la moda para cada punto, se asocian en un mismo grupo común todos aquellos puntos que han sido asociados a la misma moda.
- El algoritmo de agrupamiento LPC es un nuevo algoritmo de agrupamiento basándose en el método desarrollado por J. Einbeck, et al. [4] y realiza la búsqueda de curvas locales principales. Este nuevo algoritmo, no sólo sirve para detectar la curva, sino que también es capaz de agrupar todos los puntos pertenecientes a estas curvas y asignarlos a un mismo grupo.

Dado que apenas existen muchos estudios de comparativas de métodos de agrupamiento basados en densidades (entre otras razones por su novedad y su claro carácter aplicado) en este Proyecto de Fin de Carrera nos hemos centrado en hacer una comparación de los algoritmos DBSCAN, Mean Shift y LPC. El objetivo de esa comparación ha sido estudiar en profundidad las características de estos métodos para, de este modo, facilitar su comprensión e interpretabilidad. Está comparativa y los experimentos llevados a cabo han sido presentados en el capítulo cuatro. Teniendo en cuenta las características deseables en cualquier algoritmo de agrupamiento, se pueden extraer de la comparativa realizada los siguientes resultados:

a. *Capacidad de descubrir grupos con distinta forma*. Tanto DBSCAN como Mean Shift son capaces de descubrir grupos de formas arbitrarias, pero LPC no es capaz de descubrir grupos con formas esféricas o con una varianza frente a la curva principal muy grande.

b. *Efecto de la distribución de la densidad en los grupos*. DBSCAN y LPC son sensibles a distribuciones de densidades variables en los grupos, mientras que Mean Shift no se ve afectado por esta característica. Por el contrario, Mean Shift no funciona para grupos distribuidos uniformemente pero, DBSCAN y LPC funcionan correctamente.

c. *Capacidad de detectar ruido o outliers*. El único algoritmo capaz de detectar ruido es DBSCAN. Mean Shift y LPC no detectan la presencia de ruido en los grupos. Si se realiza la aproximación por la cual los grupos con un número de puntos inferior a un cierto umbral

se clasifican como ruido, se puede simular la detección ruido en los algoritmos Mean Shift y en LPC. De esta manera, a pesar de esta simulación para detectar ruido, se comprueba que DBSCAN se comporta mejor ante el ruido, mientras que LPC y Mean Shift son menos robustos.

d. *Escalabilidad*. DBSCAN y LPC son rápidos desde un punto de vista computacional o tiempos de cálculo del algoritmo. Por el contrario, a medida que se aumenta el número de puntos para un conjunto de datos, el tiempo de cálculo para el algoritmo Mean Shift aumenta rápidamente. Podemos decir que la escalabilidad es buena para DBSCAN y LPC, mientras que Mean Shift posee una mala escalabilidad.

e. *Parámetros de entrada*. Una ventaja fundamental de estos tres algoritmos es que ninguno necesita suponer a priori el número de grupos a encontrar. Por otro lado, DBSCAN requiere dos parámetros de entrada: el número mínimo de puntos en un área de vecindad y el radio del área de vecindad. En cambio, los algoritmos Mean Shift y LPC sólo necesitan un parámetro de entrada, el radio o ancho de banda de la ventana de desplazamiento. Esto supone una ventaja debido a que para LPC y Mean Shift sólo es necesario el ajuste de un parámetro para el proceso del aprendizaje lo que facilitará su optimización tanto en precisión como en tiempos de cálculo.

f. *Capacidad de manejar distintos tipos de datos de entrada*. Los tres algoritmos se comportan de igual manera, ya que están implementados para manejar únicamente datos numéricos.

Como conclusión general de este proyecto, podemos decir que se ha presentado un resumen actualizado del área de aprendizaje automático y más concretamente de las técnicas de agrupamiento. Hemos estudiado un nuevo tipo de técnicas de agrupamiento basadas en densidades y se han obtenido algunas de las ventajas e inconvenientes de cada una de ellas. Aunque existen parámetros que no han sido estudiados exhaustivamente como por ejemplo el radio, el estudio refleja bien las diferentes características de cada una de las técnicas. Como posibles trabajos futuros a realizar se propone la extensión del estudio a otros parámetros como el radio, o un análisis más específico sobre la robustez frente al ruido poniendo a prueba los algoritmos en otros conjuntos de datos. El claro carácter aplicado de estás técnicas hace que su uso en nuevos problemas sea uno de los motivos principales para su desarrollo. En esta línea, se propone su uso para la detección de formas en imágenes o en un entorno algo más teórico en su uso para la detección de estructuras en entornos de alta dimensionalidad como en [1].

Apéndice A

Algoritmos implementados en Matlab.

Este apéndice pretende mostrar el código en matlab de los algoritmos DBSCAN, Mean Shift Clustering y LPC.

A.1. DBSCAN.

```
% ------------------------------------------------------------------
% Function: [class,type]=dbscan(x,k,Eps)
% ------------------------------------------------------------------
% Aim:
% Clustering the data with Density-Based Scan Algorithm with
% Noise (DBSCAN)
% ------------------------------------------------------------------
% Input:
% x - data set (m,n); m-objects, n-variables
% k - number of objects in a neighborhood of an object
%     (minimal number of objects considered as a cluster)
% Eps - neighborhood radius, if not known avoid this parameter
%       or put []
% ------------------------------------------------------------------
% Output:
% class - vector specifying assignment of the i-th object to
%         certain cluster (m,1)
% type - vector specifying type of the i-th object
%        (core: 1, border: 0, outlier: -1)
% ------------------------------------------------------------------
% Example of use:
```

```
% x=[randn(30,2)*.4;randn(40,2)*.5+ones(40,1)*[4 4]];
% [class,type]=dbscan(x,5,[]);
% drawClusters2(class,x);
% --------------------------------------------------------------------
% References:
% M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based
% algorithm for discovering clusters in large spatial databases
% with noise, proc. 2nd Int. Conf. on Knowledge Discovery and
% Data Mining, Portland, OR, 1996, p. 226, available from:
% www.dbs.informatik.uni-muenchen.de/cgi-bin/papers?query=--CO
% --------------------------------------------------------------------
function [class,type]=dbscan(x,k,Eps)

[m,n]=size(x);

if nargin<3 | isempty(Eps)
   [Eps]=epsilon(x,k)
end

%Obtain matrix with euclidean distances from each point
Distances= pdist(x);
Distances= squareform(Distances);

x=[[1:m]' x];
no=1;%ClusterId, beggins with 1
touched=zeros(m,1);%All the objects are unclassified
type=zeros(1,m);
index=[1:m];%Create index with all unclassified objects

while(length(index)~=0)

    %Choose randomly an object from the data set
    value=round(rand(1)*length(index));
    if value==0
        value=1;
    end
    i=index(value);
    index(value)=[];

    if(touched(i)==0) % If object is unclassified
        ob=x(i,:);%Get the object i

        %Obtain distances from object i to the rest of objects
        D=Distances(i,:);

        %Return the indices of those elements in D smaller or
```

```
%equal than Eps
ind=find(D<=Eps);

%If the number of points is smaller than numMin is a
%border point
if length(ind)>1 & length(ind)<k+1
    type(i)=0;
    class(i)=0;
end
%If the number of points is smaller than numMin is noise
if length(ind)==1
    type(i)=-1;
    class(i)=-1;
    touched(i)=1;%Set point as classified
end
%If the number of points is bigger than numMin is a core
 %point
if length(ind)>=k+1;
    type(i)=1;
    %Set those elements in the neighbourhoud with the
    %same cluster id of the object
    class(ind)=ones(length(ind),1)*max(no);

    while ~isempty(ind)
        %Get an object from the neighbourhood of the
        %object
        ob=x(ind(1),:);
        %Set point as classified
        touched(ind(1))=1;

        %Obtain distances from object ind(1) to the rest
        %of objects
        D=Distances(ind(1),:);
        ind(1)=[];
        %Return the indices of those elements in D
        %smaller or equal than Eps
        i1=find(D<=Eps);
        if length(i1)>1
            class(i1)=no;
            if length(i1)>=k+1;

                type(ob(1))=1;%type(indice)= core point
            else
                type(ob(1))=0;%type(indice)= noise
            end
```

```
                    for i=1:length(i1)
                        %If the object is unclassified, set it as
                        %classified and append it to the list of
                        %indexes
                        if touched(i1(i))==0
                            touched(i1(i))=1;
                            ind=[ind i1(i)];
                            class(i1(i))=no;
                        end
                    end
                end
            end
            no=no+1;%Define new cluster
        end
    end
end

i1=find(class==0);
class(i1)=-1;
type(i1)=-1;

% -----------------------------------------------------------------
% Function: [Eps]=epsilon(x,k)
% -----------------------------------------------------------------
% Aim:
% Analytical way of estimating neighborhood radius for DBSCAN
% -----------------------------------------------------------------
% Input:
% x - data matrix (m,n); m-objects, n-variables
% k - number of objects in a neighborhood of an object
% (minimal number of objects considered as a cluster)

function [Eps]=epsilon(x,k)
[m,n]=size(x);

Eps=((prod(max(x)-min(x))*k*gamma(.5*n+1))/(m*sqrt(pi.^n))).^(1/n);
```

A.2. Agrupamiento Mean Shift.

```
% ------------------------------------------------------------------
% Function: [clustCent,class] = MSClusterNormal(x,bandWidth);
% ------------------------------------------------------------------
% Aim:
% Perform MeanShift Clustering of data using a flat kernel
% ------------------------------------------------------------------
% Input:
% x - data set (m,n); m-objects, n-variables
% bandWidth - is bandwidth parameter (scalar)
% ------------------------------------------------------------------
% Output:
% clustCent - mode associated to each point
% class - vector specifying assignment of the i-th object to
%  certain  cluster (m,1)
% ------------------------------------------------------------------
% Example of use:
% x=[randn(30,2)*.4;randn(40,2)*.5+ones(40,1)*[4 4]];
% [clustCent,class]=MSClusterNormal(x,5);
% drawClusters2(class,x);
% ------------------------------------------------------------------
% References:
% D. Comaniciu and P. Meer
% Mean shift: a robust approach toward feature space analysis
% Pattern Analysis and Machine Intelligence, IEEE Transactions on
% Volume 24, Issue 5, May 2002 Page(s):603 - 619
% http://dx.doi.org/10.1109/34.1000236
% ------------------------------------------------------------------
function [clustCent,class] = MSClusterNormal(x,bandWidth)

[m,n] = size(x);
clusterId = 0 ; %Cluster identifier
%All the objects have initially as center the origin
clustCent = zeros(m,n);
%Define index with objtects that are not classified
index = [1:m];
h = bandWidth;
%All the objects are unclassified
touched = zeros(m,1);

while(~isempty(index))

    %Choose a random point to classify it
```

```
    value = round(rand*length(index));
    if value == 0
        value = 1;
    end
    i=index(value);

    %Find mode
    [peak, indcpts] = findModeOpt(x,i,h);

    %Find closest point to center found and set is as center
    indCenter = findClosestPoint(x,peak);

    %Obtain indexes of points with same clusterCenter (ind)
    [c,ind,ib] = intersect(clustCent,x(indCenter,:),'rows');

    %If cluster is already defined store it and points in h/4
    %radious of path
    if length(ind) == 1

       class(indcpts) = class(ind(1));
       %Store center Id
       clustCent(indcpts,:) = repmat(x(indCenter,:),length(indcpts),1);

    %If new cluster, create new cluster id and store it and points
    % in mode's neighbourhood and points in h/4 radious of path
    else

        clusterId=clusterId+1;
        %Find points in the neighbourhood of the peak
        indcpts2 = find(h>sqrt(sum((x-repmat(peak,size(x,1),1)).^2,2)));
        class(indcpts) = clusterId;
        class(indcpts2) = clusterId;
        %Store center Ids
        clustCent(indcpts,:) = repmat(x(indCenter,:),length(indcpts),1);
        %Store center Ids
        clustCent(indcpts2,:) = repmat(x(indCenter,:),length(indcpts2),1);

    end

    %Delete from index all previously classified values
    index = setdiff(index,indcpts);
    index = setdiff(index,indcpts2);

end

i1=find(class==0);
```

```
class(i1)=-1;

% -----------------------------------------------------------------
% function [mode, indcpts] = findModeOpt(x,i,h)
% -----------------------------------------------------------------
% Aim:
% Compute mean shifts until a mode is found
% -----------------------------------------------------------------
% Input:
% x - m x n matrix containing m data points with n dimensions
% i - index of the data point for which we wish to compute its
%     associated mode;
% h - search window radius (bandwidth)
% -----------------------------------------------------------------
% Output:
% mode - local maxima of the p.d.f
% indcpts - index containing locations of vectors within a
%           distance h/4 from the path
% -----------------------------------------------------------------

function [mode, indcpts] = findModeOpt(x,i,h)

[m,n] = size(x);
%Vector storing a 1 for each point that is within a distance h/4
%from the path, and a 0 otherwise
cpts = zeros(1,m);
cpts(i) = 1;
stopThresh = 1e-10*ones(1,n); %Stop the gradient at this threshold
xt = x(i,:); %Start point

mean_shift = computeMean(xt,x,h); %Compute mean shift

while abs(mean_shift) > stopThresh

    xt = xt + mean_shift; %translate the window
    mean_shift = computeMean(xt,x,h); %Compute mean shift

    %Find points within a distance h/4 from the path and set it to 1
    aux = find((h/4)>sqrt(sum((x-repmat(xt,size(x,1),1)).^2,2)));
    cpts(aux) = 1;

end

indcpts = find(cpts==1);
mode = xt + mean_shift;
```

```
% ------------------------------------------------------------------
% function [mean]= computeMean(xt,x,h)
% ------------------------------------------------------------------
% Aim:
% Compute mean shift
% ------------------------------------------------------------------
% Input:
% xt - data point to be shifted
% x - m x n matrix containing m data points with n dimensions
% h - search window radius (bandwidth)
% ------------------------------------------------------------------
% Output:
% mean - computed mean shift
% ------------------------------------------------------------------

function [mean]= computeMean(xt,x,h)


[m,n] = size(x);
a=zeros(1,n);
b=0;

for i=1:m
   xi = x(i,:);
   a = a + xi * exp( - ( norm((xt-xi)/h) ^2) / 2);
   b = b + exp( - ( norm((xt-xi)/h) ^2) / 2);
end

mean = a / b - xt;


% ------------------------------------------------------------------
% function [index] = findClosestPoint(x,xt)
% ------------------------------------------------------------------
% Aim:
% Find closest neighbour of a point
% ------------------------------------------------------------------
% Input:
% x - m x n matrix containing m data points in n dimensions
% xt - data point from which we want to find its closest neighbour
% ------------------------------------------------------------------
% Output:
% index - index of closest point of xt in data points x

function [index] = findClosestPoint(x,xt)
```

```
[m,n] = size(x);
D = sqrt(sum((((repmat(xt,m,1))-x).^2)'));%Compute distances

%Find index of closest point
index = find(D==min(D));
```

A.3. LPC.

```
% ------------------------------------------------------------------
% Function: [class,curves]=LPC(x,h)
% ------------------------------------------------------------------
% Aim:
% Clustering the data by using Local principal curves
% ------------------------------------------------------------------
% Input:
% x - data set (m,n); m-objects, n-variables
% h - neighborhood radius, bandwitdh
% ------------------------------------------------------------------
% Output:
% class - vector specifying assignment of the i-th object to
%         certain cluster (m,1)
% curves - vector containing the objects that define each curves
% ------------------------------------------------------------------
% Example of use:
% x = -5:1/20:5;
% y = x.^2+randn(size(x))
% m = [(1:1/20:11)' y']
% m = [(m-min(m))./(max(m)-min(m))];
% [class,curves]=LPC(x,0.05)
% drawClusters2(class,x)
% ------------------------------------------------------------------
% References:
% J. Einbeck, G. Tutz, L.Evers, Local Principal Curves.
% Sonderforschungsbereich 386, Paper 320 (2003)
% Online unter: http://epub.ub.uni-muenchen.de/

function [class,curves]=LPC(x,h)

[m,n] = size(x);

clustId = 1;
touched = zeros(m,1);%All the objects are unclassified
index = [1:m];
%Stop computing means at this threshold
Thr = h*1e-5*ones(1,n);
curves = [];
class = -1*ones(m,1);
%  plot(x(:,1),x(:,2),'*')
%  hold on
```

```
while(~isempty(index))

    value=round(rand(1)*length(index));
    if value==0
        value=1;
    end
    i=index(value);
    index(value)=[];

    if touched(i)==0;%If the object i is unclassified

        m=x(i,:);%Get the object i
        curves(length(curves)+1,:) = [clustId m];

        touched(i) = 1;
        %Find objects in the neighbourhood of object i
        ind = find(h > sqrt(sum((x-repmat(m,size(x,1),1)).^2,2)));
        %Check if objects in neighbourhood where previously
        %classified
        s = length(ind);
        a = find(class(ind)==-1);
        if(length(a)< s/3)
            b = find(class(ind)==-1);
            class(ind(a)) = class(b(1));
            touched(ind) = 1;
            ind = find(touched==1);
            index = setdiff(index,ind);
        else
            %Set those elements in the neighbourhoud with the
            %same cluster id of the object
            class(ind)=ones(length(ind),1)*clustId;
            touched(ind) = 1;

            mu_x = computeMean(m,x,h);
            mu_x2 = 10*mu_x;
            mu_x3 = 10*mu_x;
            gamma_x2 = [];

            mu_x_aux = mu_x; %Save parameter to run backwards later
            ind_aux = ind;
            m_aux = m;


            while((abs(mu_x2-mu_x)> Thr)&(abs(mu_x3-mu_x)>Thr))

              %Calculate covariance matrix
```

```
                covMat = cov(x(ind,:));
                %Calculate eigenvectors and eigenvalues of covMat
                [Vx,Ex] = eig(covMat);
                gamma_x = Vx(:,length(Vx));
                if ~isempty(gamma_x2)
                  cos_alpha = gamma_x'*gamma_x2;
                  if cos_alpha < 0
                     gamma_x = - gamma_x;
                  end
                end

                m = mu_x + h*gamma_x';
                curves(length(curves)+1,:) = [clustId m];

                ind = find(h>sqrt(sum((x-repmat(m,size(x,1),1)).^2,2)));

                if(~isempty(find(class(ind)==-1)))
                    touched(ind) = 1;
                    %Set those elements in the neighbourhood
                    % with the same cluster id of the object
                    class(ind)=ones(length(ind),1)*clustId;
                      mu_x3 = mu_x2;
                      mu_x2 = mu_x;
                      mu_x = computeMean(m,x,h);
                      gamma_x2 = gamma_x;
                  else
                      mu_x2 = mu_x;
                  end

              end

              %Run backwards
              mu_x = mu_x_aux;
              mu_x2 = 10*mu_x;
              mu_x3 = 10*mu_x;
              gamma_x2 = [];
              ind = ind_aux;
              m = m_aux;


              while((abs(mu_x2-mu_x)>Thr)&(abs(mu_x3-mu_x)>Thr))

%Calculate the covariance matrix of neighbourhood
                  covMat = cov(x(ind,:));
                  %Calculate eigenvectors and eigenvalues of covMat
                  [Vx,Ex] = eig(covMat);
```

```
                gamma_x = - Vx(:,length(Vx));
                if ~isempty(gamma_x2)
                  cos_alpha = gamma_x'*gamma_x2;
                  if cos_alpha < 0
                      gamma_x = - gamma_x;
                  end
                end

                m = mu_x + h*gamma_x';
                curves(length(curves)+1,:) = [clustId m];

                ind = find(h > sqrt(sum((x-repmat(m,size(x,1),1)).^2,2)));
                if(~isempty(find(class(ind)==-1)))

                    touched(ind) = 1;
                    %Set those elements in the neighbourhoud with
                    %the same cluster id of the object
                    class(ind)=ones(length(ind),1)*clustId;
                    mu_x3 = mu_x2;
                    mu_x2 = mu_x;
                    mu_x = computeMean(m,x,h);
                    gamma_x2 = gamma_x;
                else
                    mu_x2 = mu_x;
                end


            end

            clustId = clustId +1;
            ind = find(touched==1);
            index = setdiff(index,ind);
        end
    end

end


% --------------------------------------------------------------------
% function [mean]= computeMean(xt,x,h)
% --------------------------------------------------------------------
% Aim:
% Compute mean shift
% --------------------------------------------------------------------
% Input:
% xt - data point to be shifted
```

```
% x - m x n matrix containing m data points, each point has n dim
% h - search window radius (bandwidth)
% -----------------------------------------------------------------
% Output:
% mean - computed mean shift

function [mean]= computeMean(xt,x,h)


[m,n] = size(x);
a=zeros(1,n);
b=0;

for i=1:m
   xi = x(i,:);
   a = a + xi * exp( - ( norm((xt-xi)/h) ^2) / 2);
   b = b + exp( - ( norm((xt-xi)/h) ^2) / 2);
end

mean = a / b;
```

Bibliografía

[1] L. Yu, C. Ding, and S. Loscalzo, "Stable feature selection via dense feature groups," in *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 803–811, ACM, 2008.

[2] M. Ester, H.-P. Kriegel, S. Jörg, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," 1996.

[3] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603–619, 2002.

[4] J. Einbeck, G. Tutz, and L. Evers, "Local principal curves," tech. rep., June 2007.

[5] T. M. Mitchell, *Machine Learning*. McGraw-Hill Science/Engineering/Math, March 1997.

[6] P. Isasi Viñuela and I. M. Galván León, *Redes de neuronas artificiales: un enfoque práctico*. Monographs and Textbooks on Probability and Mathematical Statistics, Madrid: Prentice Hall, 2004.

[7] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, November 1995.

[8] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[9] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, March 1986.

[10] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 21, no. 3, pp. 660–674, 1991.

[11] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, Third Edition*. Academic Press, 2006.

[12] K. Fukunaga, *Introduction to Statistical Pattern Recognition, Second Edition (Computer Science and Scientific Computing Series)*. Academic Press, September 1990.

[13] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, August 2006.

[14] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000.

[15] C. Cortes and V. Vapnik, "Support vector networks," in *Machine Learning*, pp. 273–297, 1995.

[16] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[17] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, (New York, NY, USA), pp. 92–100, ACM Press, 1998.

[18] T. M. Mitchell, "The role of unlabeled data in supervised learning," in *In Proceedings of the Sixth International Colloquium on Cognitive Science*, 1999.

[19] O. Chapelle, B. Schölkopf, and A. Zien, eds., *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2006.

[20] A. N. Langville and C. D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, July 2006.

[21] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.

[22] Y. Kim, W. N. Street, and F. Menczer, "Feature selection in data mining," in *Data mining: opportunities and challenges*, (Hershey, PA, USA), pp. 80–105, IGI Publishing, 2003.

[23] F. Attneave, "Dimensions of similarity," *American Journal of Psychology*, vol. 63, no. 4, pp. 516–556, 1950.

[24] S. S. Member and R. J. Fellow, "Similarity measures," 1999.

[25] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. Wiley Series in Probability and Mathematical Statistics.Applied Probability and Statistics, New York: Wiley, 1990.

[26] B. S. Everitt, S. Landau, and M. Leese, *Cluster Analysis*. Arnold Publishers, May 2001.

[27] W. tau Yih and C. Meek, "Improving similarity measures for short segments of text.," in *AAAI*, pp. 1489–1494, AAAI Press, 2007.

[28] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 4–37, January 2000.

[29] J. B. Macqueen, "Some methods for classification and analysis of multivariate observations," in *Procedings of the Fifth Berkeley Symposium on Math, Statistics, and Probability*, vol. 1, pp. 281–297, University of California Press, 1967.

[30] K. Stoffel and A. Belkoniene, "Parallel k/h-means clustering for large data sets," in *Euro-Par '99: Proceedings of the 5th International Euro-Par Conference on Parallel Processing*, (London, UK), pp. 1451–1454, Springer-Verlag, 1999.

[31] J. M. Peña, J. A. Lozano, and P. Larrañaga, "An empirical comparison of four initialization methods for the k-means algorithm," *Pattern Recogn. Lett.*, vol. 20, no. 10, pp. 1027–1040, 1999.

[32] G. H. Ball and D. J. Hall, "A clustering technique for summarizing multivariate data," *Behavioral Sci*, vol. 12, pp. 153–155, 1967.

[33] V. Estivill-Castro and J. Yang, "Fast and robust general purpose clustering algorithms," *Data Min. Knowl. Discov.*, vol. 8, no. 2, pp. 127–150, 2004.

[34] P. H. Salinas, V. J. Albornoz, P. A. Reyes, B. M. Erazo, and V. R. Ide, "Análisis de componentes principales aplicado a variables respecto a la mujer gestante en la región de las américas," *Rev. chil. obstet. ginecol.*, vol. 71, no. 1, pp. 17–25, 2006.

[35] S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," in *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 73–84, ACM, 1998.

[36] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *SIGMOD '96*, pp. 103–114, 1996.

[37] S. Guha, R. Rastogi, and K. Shim, "Rock: A robust clustering algorithm for categorical attributes," in *Information Systems*, pp. 512–521, 1999.

[38] G. J. Mclachlan and T. Krishnan, *The EM Algorithm and Extensions*. John Wiley and Sons, 1997.

[39] B. L. Tran T.N., Wehrens R., "Knn density-based clustering for high-dimensional multispectral images," 2003.

[40] L. Ertoz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *SIAM International Conference on Data Mining*, vol. 47, 2003.

[41] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander, "A distribution-based clustering algorithm for mining in large spatial databases," *Data Engineering, International Conference on*, vol. 0, p. 324, 1998.

[42] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *Knowledge Discovery and Data Mining*, pp. 58–65, 1998.

[43] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," *SIGMOD Rec.*, vol. 28, no. 2, pp. 49–60, 1999.

[44] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm gdbscan and its applications," *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 169–194, 1998.

[45] L. Hermes, T. Zöller, J. M. Buhmann, R. Friedrich, and W. Universität, "Parametric distributional clustering for image segmentation," in *In Proc. of European Conference on Computer Vision (ECCV), 2002*, pp. 577–591, Springer, 2002.

[46] C. Böhm, K. Kailing, P. Kröger, and A. Zimek, "Computing clusters of correlation connected objects," in *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 455–466, ACM, 2004.

[47] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603–619, 2002.

[48] W. Wang, J. Yang, and R. R. Muntz, "Sting: A statistical information grid approach to spatial data mining," in *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece* (M. Jarke, M. J. Carey, K. R. Dittrich, F. H. Lochovsky, P. Loucopoulos, and M. A. Jeusfeld, eds.), pp. 186–195, Morgan Kaufmann, 1997.

[49] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "Wavecluster: A multi-resolution clustering approach for very large spatial databases," in *VLDB*, pp. 428–439, 1998.

[50] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," in *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA* (L. M. Haas and A. Tiwary, eds.), pp. 94–105, ACM Press, 1998.

[51] S. Goil, H. Nagesh, and A. Choudhary, "Mafia: Efficient and scalable subspace clustering for very large data sets," Tech. Rep. 9906-010, Northwestern University, June 1999.

[52] W. keng Liao, Y. Liu, and A. Choudhary, "A grid-based clustering algorithm using adaptive mesh refinement," 2008.

[53] H. Kriegel, "Density-based cluster- and outlier analysis." Website, 2000. `http://www.dbs.informatik.uni-muenchen.de/Forschung/KDD/Clustering/index.html`.

[54] I. T. Jolliffe, *Principal component analysis*. Springer New York, 2002.

[55] T. Hastie and W. Stützle, "Principal curves," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 502–516, 1989.

[56] R. Tibshirani, "Principal curves revisited," *Statistics and Computing*, vol. 2, pp. 183–190, 1992.

[57] B. Kégl, A. Krzyzak, T. Linder, and K. Zeger, "Learning and design of principal curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 3, pp. 281–297, 2000.

[58] P. Delicado, "Another look at principal curves and surfaces," *Journal of Multivariate Analysis*, vol. 77, pp. 84–116, 2001.

[59] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2 ed., 2005.

[60] B. H. K. D. H., "Neural networks in materials science," *ISIJ International*, vol. 39, p. 966979, 1999.

[61] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, pp. 264–323, 1999.

[62] R. Xu and D. Wunsch, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, no. 3, pp. 645–678, 2005.

[63] J. A. K. Suykens, "Nonlinear modeling and support vector machines," *IEEE Instrumentation and Measurement Technology Conference, Budapest*, pp. 287–294, 2001.

[64] M. R. Anderberg, *Cluster Analysis for Applications*. Monographs and Textbooks on Probability and Mathematical Statistics, New York: Academic Press, Inc., 1973.

[65] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. ADDISON WESLEY PUBLI, June 2006.

[66] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," *Informatica*, vol. 31, no. 3, pp. 249–268, 2007.

[67] X. Zhu, "Semi-supervised learning literature survey," Tech. Rep. 1530, Computer Sciences, University of Wisconsin-Madison, 2005.